VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

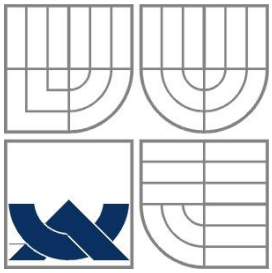
ITIL V PROSTŘEDÍ AGILNÍHO VÝVOJE SOFTWARE METODIKOU SCRUM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

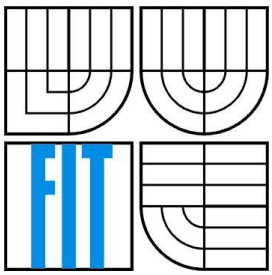
AUTOR PRÁCE
AUTHOR

Bc. MONIKA KUBALCOVÁ

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

ITIL V PROSTŘEDÍ AGILNÍHO VÝVOJE SOFTWARE METODIKOU SCRUM

ITIL IN THE CONTEXT OF AGILE SOFTWARE DEVELOPMENT METHODOLOGY SCRUM

DIPLOMOVÁ PRÁCE
MASTER‘S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MONIKA KUBALCOVÁ

VEDOUCÍ PRÁCE
SUPERVISOR

RNDr. MAREK RYCHLÝ, Ph.D.

BRNO 2014

Abstrakt

Tato práce se zabývá problematikou integrace procesní knihovny ITIL a agilní metodiky Scrum ve společnosti vyvíjející IT služby. Po teoretickém popisu obou přístupů a analýzy procesů ve vybrané společnosti následuje navržení metodiky, která zajistí provázání obou procesních přístupů a jejich bezproblémové fungování. Jako podpora metodiky je implementována webová aplikace, pomocí které lze vyhodnotit efektivitu navržených procesů vykreslením důležitých informací do grafů. Navržená metodika byla prezentována ve společnosti AVG Technologies, která využila její navržené procesy jako předlohu pro popis svých interních procesů. Výsledky této práce poskytují společnostem, které ve svém vývoji chtějí použít Scrum společně s procesy z ITIL, předlohu v podobě navržených procesů.

Abstract

This thesis follows concept of integration ITIL library and agile methodology Scrum into the IT company developing IT services. After theoretical analysis of both concepts and analysis of chosen company, we will develope methodology, which will tie both process approaches and ensure their perfect synergy. As a methodology support, web application was developed. It evaluates effectivness of processes using key informations and their plot representation. Proposed methodology was presented in the AVG Technologies, which used described processes as template for its internal proceses. Main result of the thesis is set of templates describing processes, which could be specified and applied in companies using Scrum together with ITIL for their development.

Klíčová slova

ITIL, modelování procesů, agilní metodiky, Scrum, BPMN

Keywords

ITIL, business process modelling, agile methodologies, Scrum, BPMN

Citace

Kubalcová Monika: ITIL v prostředí agilního vývoje software metodikou Scrum, diplomová práce, Brno, FIT VUT v Brně, 2014

ITIL v prostředí agilního vývoje software metodikou Scrum

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením RNDr. Marka Rychlého, Ph.D.
Další informace mi poskytli Mgr. Lukáš Havlíček, Bc. Robert Zeman, Ing. Milan Tomášek a Ing. Karel Smutný.
Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

……
Monika Kubalcová
27. 05. 2014

Poděkování

Na tomto místě bych ráda poděkovala RNDr. Marku Rychlému, Ph.D., který mě v práci vedl a poskytl mi cenné rady a konzultace při řešení mé práce. Též bych chtěla poděkovat Mgr. Lukáši Havlíčkovi, Bc. Robertu Zemanovi, Ing. Milanu Tomáškovi a Ing. Karlu Smutnému z firmy AVG Technologies, za poskytnutí informací a osobních zkušeností při konzultování k tématu mé práce.

Rovněž děkuji rodině za podporu během celého studia.

Obsah

1 Úvod

Na počátku 21. století, se svět vyvíjí velmi dynamicky, s obrovským množstvím změn, které přicházející za chodu. Změny se velmi rychle objevují hlavně v technických možnostech díky lepšímu technologickému vybavení, lepší dostupnosti výpočetní techniky jak pro velké společnosti, tak pro širokou veřejnost, většímu množství specialistů v oblasti informačních technologií a kvalitnějším vývojovým prostředím. Těmto změnám musí reflektovat i rychlost vývoje software. Protože software dnes může vyvíjet prakticky kdokoliv na to dostatečně vzdělaný a schopný, je jasné, že množství nových nápadů vzniká velmi rychle, čímž konkurence neustále stoupá. Vyvíjí-li společnost aplikaci příliš dlouho, je jisté, že konkurenti mezitím vyvinou stejnou, ne-li lepší a rozhodně v lepším čase.

Kvalita vývoje software se tak stává čím dál tím více důležitější, protože přímo ovlivňuje obchodní strategie organizací vyvíjející software. Pokud vývoj trvá příliš dlouho, není úspěšný a je na něj třeba investovat mnoho zdrojů, může to mít fatální následky pro celou organizaci. Naštěstí dnes existují takové návody a metodiky, které organizace učí a doporučují jim, jak přizpůsobit strukturu a chod společnosti pro docílení co nejoptimálnějšího a nejúspěšnějšího vývoje, dodání kvalitních produktů zákazníkům a udržení tak dobrého postavení organizace na trhu. Jedním z často implementovaných přístupů je ITIL. Je to soubor návodů a postupů, poskytující sadu procesů pro organizace všech velikostí, které jim při správném použití zaručí lepší plánování a využívání zdrojů, efektivní návrh procesů a činností, určení důležitých rolí, návrh struktury organizace a celkové lepší dodání služby zákazníkovi. Implementace knihovny ITIL není zcela bezplatná, soubor pěti publikací je nutno zakoupit, podle nich je možné s pomocí proškolených lidí organizaci přetvářet na procesně zaměřenou. V této fázi je velmi důležité, jak kvalitně bude knihovna implementována a procesy z ní budou fungovat pro konkrétní organizaci, protože právě na tom bude záviset další úspěch organizace. Nutno podotknout, že není třeba krok pro kroku dodržet všechny rady v publikacích ITIL, je lepší je vhodně modifikovat pro použití konkrétní organizace.

Na druhé straně existují také metodiky a postupy, které se zabývají přímo jen vývojem software. Tradiční metodiky z 50. let postupně střídají nové, moderní metodiky, které lépe reagují na dnešní podmínky vývoje − řeč je o agilních metodikách. Jejich hlavními rysy jsou redukce nepotřebných dokumentací a procesů a upřednostněním interakce mezi členy týmu a spolupráce se zákazníkem.

Velké množství firem vyvíjející software a poskytující služby zákazníkům je dnes procesně založeno právě podle rámce ITIL. Ten může být v některých ohledech pro organizace a její zaměstnance zbytečně svazující, jeho pravidla mohou být příliš robustní, množství dokumentace a administrativy velké a může se stát, že samotný vývojový cyklus se tak pod tíhou ostatních povinností stane neefektivní a zdlouhavý. Pokud je navíc vývoj řízen podle jedné z tradičních metodik, může délka a kvalita zpracování být velmi kritická. Zavádění agilních metodik do procesně fungujících firem je tak stále častější.

Během posledního roku a půl magisterského studia na Fakultě informačních technologií Vysokého učení technického v Brně jsem měla šanci pracovat ve firmě AVG Technologies, která se zabývá vývojem antivirů a služeb s nimi spojenými. V oddělení vyvíjejícím webové stránky pro podporu prodeje a údržby nabízených služeb jsem v oddělení Release managementu pomáhala plánovat a koordinovat vydávání releasů. Celé oddělení vývoje webů je již několik let procesně založeno podle knihovny ITIL a mimo to se během posledního roku rozhodlo zefektivnit vývoj software zavedením agilní metodiky Scrum. Protože jsem měla šanci sledovat fungování před i po zavedením Scrum do oddělení a vidět tak transformaci týmů i postupů, rozhodla jsem se na toto téma zpracovat svou

diplomovou práci. Jejím cílem je analyzovat knihovnu ITIL i agilní metodiku Scrum a zjistit, jestli mohou společně fungovat pro efektivní dodávání software. Výstupem této práce je navržení metodiky SCRIL, která popisuje spojení obou přístupů. Metodika je doplněna o návrh webové aplikace, která podporuje monitorování spojení obou metodik.

Kapitola 2 stručně popisuje tradiční vývoje metodiky software, pro vytvoření představy čtenáře, jakým způsobem se software vyvíjí bez použití agilních metodik a lepší pochopení důvodu zavádění agilních metodik.

Kapitola 3 popisuje vznik a důležité vlastnosti agilních metodik obecně, v kapitole 4 se pak konkrétně dále budeme zabývat jen agilní metodikou Scrum.

Kapitola 5 popisuje procesní knihovnu ITIL a definice jejích nejdůležitějších procesů, kapitola 6 pak nástrojům a možnostem modelování podnikových procesů.

Kapitola 7 se zabývá analýzou a popisem integrace agilní metodiky Scrum v prostředí procesů z procesní knihovny ITIL ve společnosti AVG Technologies, těsně po spojení obou metodik. Rovněž jsou popsány i problémy, se kterými se společnost při integraci setkala.

Kapitola 8 popisuje návrh metodiky SCRIL, která se snaží co nejlépe popsat procesy, které se nachází v bodě spojení obou metodik spolu s jejich vizualizací. Procesy jsou navrženy tak, aby se daly použít jako základní návrh procesů v jakékoliv společnosti dodávající IT služby zákazníkům.

Kapitola 9 obsahuje popis navržené webové aplikace, která umožňuje automatickou generaci grafů (i pro pravidelný reporting) pro snadné zhodnocení spojení obou metodik a pro udržení procesů efektivními.

Kapitola 10 hodnotí průběh integrace metodiky Scrum do prostředí ITIL ve společnosti AVG Technologies po dalším půlroce používání a zhodnocuje navrženou metodiku SCRIL z pohledu zkušeností z praxe.

1.1 Pojmy

V této práci budeme často používat pojmy, které mohou nabývat různých významů v různých kontextech a čtenáři jejich význam a hlavně jejich rozdíl, nemusí být na první pohled úplně jasný. Proto bude lepší si pojmy jednoznačně pro naše účely definovat.

Metodologie - vědní disciplína, která se zabývá metodami, jejich tvorbou a aplikacemi [21].

Metodika - souhrn postupů, pravidel a doporučení, který vedou k řešení nějakého problému [21].

Metoda - soustavný postup nebo návod, který v dané oblasti vede k cíli, v ideálním případě nezávisle na schopnostech toho, kdo ho provádí [22].

Proces - je soubor vzájemně souvisejících nebo vzájemně působících činností, které přeměňují vstupy na výstupy. Každý proces má svého vlastníka, vstupy, aktivity a výstupy [ISO 10006].

Služba - je prostředek poskytování hodnoty zákazníkovi, usnadnění výstupů, které chce dosáhnout bez vystavení rizikům a specifickým nákladům [18].

2 Softwarové inženýrství

V této kapitole se seznámíme se vznikem odvětví softwarového inženýrství a vznikem modelů životních cyklů vývoje software, které jsou dnes známé především jako „tradiční metodiky“ pro vývoj software. To je důležité především proto, aby si čtenář vytvořil představu o tom, jak vývoj software dříve fungoval, co na něm bylo špatného, proč začaly vznikat nové metodiky a jak se oproti těm starým liší.

Software je dnes běžnou součástí našich životů, svěřujeme mu naše peníze, informace a někdy i naše zdraví a životy. Je tedy jasné, že požadavky kladené na jeho správnost, spolehlivost a bezpečnost se pořád zvyšují. Na druhou stranu s větší kvalitou stoupá i čas strávený vývojem softwaru, od čehož se neblaze odvíjí i jeho cena. Opovědí na to, jak ekonomicky vyvíjet bezpečný, potřebný a kvalitní software je právě softwarové inženýrství.

„*Softwarové inženýrství je zavedení a používání řádných inženýrských principů tak, abychom dosáhli ekonomické tvorby softwaru, který je spolehlivý a pracuje účinně na dostupných výpočetních prostředcích*.“ (Friedrich Bauer, konference NATO 1968)

Z definice vyplývá, že softwarové inženýrství není jenom samotná implementace hotového software, jak si většina lidí může myslet, ale je to komplexní pohled na celý proces vývoje od specifikace požadavků na zadání až po nasazení a údržbu u zákazníka.

2.1 Historie

Vznik softwarového inženýrství můžeme datovat na 50. a 60. léta 20. století, kdy začala klesat pořizovací cena hardware. Z velkých sálových počítačů, ovládaných jednoduchým programem, se kterým většinou uměl pracovat pouze jeho tvůrce, se postupně staly menší počítače za lepší cenu a tak poptávka po software začala stoupat. Najednou bylo třeba tvorby programů větších rozměrů, s čímž začaly přicházet problémy, které vyústily až k zavedení pojmu softwarové krize. Ta se projevovala protahováním a prodražováním projektů, špatnou kvalitou výsledných produktů a problematickou údržbou. V 70. letech 20. století snaha o překonání krize vedla ke hledání lepších programovacích praktik použitelných v praxi a zkoumání životního cyklu tvorby softwaru zaměřeného na kvalitu výsledného produktu. Na konci 70. let se v rámci životního cyklu začal klást větší důraz na prvotní fáze, tedy analýzu, specifikaci požadavků a návrh, zatímco v 80. letech se pozornost přesunula na etapy nasazení software a jeho údržby. Vznikají první vývojová prostředí, vyvíjí se logické a funkcionální programování a velký rozmach zažívají objektově orientované přístupy, analýzy a návrhy. V 90. letech se zavádějí metriky pro posouzení kvality softwarového procesu, software se vyvíjí pomocí prototypování s důrazem na pozdější znovu použitelnost [12].

2.2 Životní cykly softwaru

Nyní si popíšeme pár tradičních, lety prověřených metodik a modelů vývoje softwaru. Tento stručný přehled bude potřebný pro lepší možnost srovnání těchto metodik s novými přístupy, které jsou hlavním tématem práce [11][12].

2.2.1 Vodopádový model

Pravděpodobně nejtradičnějším modelem životního cyklu softwaru je právě vodopádový model. Dnes bývá zmiňován hlavně v souvislosti s tím, jak by se software vyvíjet neměl, ale pro jednodušší projekty může být za určitých podmínek dobře použitelný i v současné době. V 70. letech, když vznikal, přinesl jako první členění na jednotlivé fáze softwarového procesu a jejich logické řazení, ze kterého se vychází dodnes. Jeho výhodou je jednoduchost a poměrně snadné manažerské řízení.

Jednotlivé fáze jsou řazeny za sebou a každá další fáze může začít až po skončení předcházející. Největším problémem tohoto modelu je to, že zákazník na začátku v podstatě není schopen přesně definovat své požadavky na výsledný systém, ty se během vývoje mohou změnit. Protože ale dostane první funkční verzi produktu až na konci cyklu modelu, kde může zjistit, že vlastně úplně nevyhovuje jeho požadavkům, musí se vývoj vrátit až na začátek cyklu a všechny fáze probíhají znovu. To je velmi nákladné jak na čas, tak na prostředky. Zákazník se projektu účastní tedy jen na začátku, při definici požadavků a na konci, kdy přebírá hotový produkt. Příklad vodopádového modelu zobrazuje Příloha A (*Obrázek 26*).

2.2.2 Spirálový model

Další z modelů životního cyklu softwaru je mnohem sofistikovanější a komplikovanější, než model vodopádový. Obsahuje všechny jeho fáze vývoje obohacené o fázi analýzu rizik, kterou skrze celý cyklus několikrát opakuje. Oproti vodopádovému modelu přidává další důležitý přístup − iterativní. Spirálový model obsahuje několik iterací sekvenčního průběhu, kdy každá z iterací zpřesní požadavky na systém, uvažuje nová rizika a dá zákazníkovi jistý výstup v podobě tzv. prototypu. Prototyp je verze výsledného softwaru s omezenou funkcionalitou, která se po použití zahodí a v dalším cyklu se software (nebo další prototyp) vytváří znovu, díky čemuž se na chyby přichází průběžně a ne až na konci. Rovněž model eliminuje problém nepřesných nebo neúplných požadavků od zákazníka, protože je schopný tyto změny zpracovat v průběhu dalšího cyklu.

Během celého cyklu je kladen velký důraz na plánování dalších postupů – vytváření požadavků, plán použití metod, integrace, testování, atd. Na konci každého cyklu se vytváří plán pro cyklus další. Analýza rizik, jak již bylo řečeno, probíhá v rámci každého z cyklů spirály a probíhá na vysoké odborné úrovni. Mezi rizika můžeme zahrnout všechny události, které eventuálně mohou vést k nedodržení zadaných cílů, jejich analýza má proto za úkol připravit reakce na tyto události. Model je vhodný i pro rozsáhlejší projekty, vyžaduje však precizní kontroly výstupů jednotlivých fází a dobré zkušenosti členů týmu, které při nedostačující kvalitě mohou způsobit krach celého projektu. Příklad Spirálového modelu zobrazuje Příloha A *(Obrázek 27).*

2.2.3 Metodika RUP

Rational Unified Process (RUP) je výsledek výzkumu koordinovaný společností Rational, který se oproti koncepčním návrhům spirálového a vodopádového modelu reálně používá v praxi. Metodika, výrazně objektově orientovaná, je chápána jako framework (softwarová struktura pro podporu programování obsahující pomocné nástroje, dokumenty, podporu pro návrhové vzory atp.), který má být uzpůsoben konkrétním projektům. Díky zmíněné robustnosti může být nasazení do dané společnosti dost náročné, což je jedna z hlavních nevýhod tohoto přístupu k vývoji software.

Důraz je kladen na vizualizaci softwarového systému pomocí jazyka UML (*„Unified Modelling Language“*), která pomáhá zobrazit systém jako celek a usnadňuje komunikace mezi členy vývojového

týmu. Metodika je založena na systému řízení změn a na vývoji softwarového produktu iterativním způsobem s využitím již existujících komponent a spadá do skupiny tzv. přístupů řízených případy použití (případ použití je definován jako funkce, kterou systém vykonává jménem jednotlivých účastníků systému nebo v jejich prospěch).

RUP obsahuje čtyři základní fáze vývoje (zahájení, rozpracování, konstrukce a zavedení), které jsou opakovány v několika iteracích, před začátkem další iteraci musí být splněna kritéria předchozí iterace. Ve fázi zahájení se definují účel a rozsah projektu, ve fázi rozpracování se definuje základ architektury projektu a analyzují se požadavky uživatele, fáze konstrukce se zabývá především vývojem designu a zdrojových kódů aplikace a poslední fáze, zavedení, předáním výsledného projektu uživateli nebo překlopení do dalšího vývojového cyklu. Příklad struktury RUP metodiky zobrazuje Příloha A *(Obrázek 28).*

2.2.4 Shrnutí

V předešlém textu jsme si postupně popsali nejznámější z modelů a metodik vývoje softwaru, které řadíme k těm „klasickým“, léty prověřeným přístupům. Z jejich popisu vyplývá, že velká většina z nich se dnes pro určité typy problému různých velikostí projektů nedá použít, přestože se tak stále děje. Hlavně proto postupně začaly vznikat metodiky a techniky, které díky svému inovativnímu přístupu k vývoji dokáží software vyvinout efektivněji a rychleji za spotřeby méně prostředků. Řeč je o agilních metodikách, které budou hlavním tématem další kapitoly.

3 Agilní metodiky

V době používání tradičních metodik vývoje softwaru, jejichž popisu je věnována předcházející kapitola, se dával velký důraz na kvalitu výsledného řešení. Protože v té době konkurence nebyla tak velká a vývojem se zabývalo jen pár větších firem a specialistů, zákazník si radši počkal, aby dostal kvalitnější výsledek. Naopak dnes se zákazník spokojí s méně kvalitním řešením, hlavně když na něj nebude muset dlouho čekat, tedy se zákazníkovi předá v kratším čase neúplný výsledek, který se postupně doplňuje a vylepšuje na základě doplňujících požadavků. Kritickým faktorem dodávání software se tak stal čas.

Tak např. při aplikaci vodopádového modelu životního cyklu software, který postupně prochází všemi svými fázemi od provedení analýzy a návrhu až po otestování a nasazení, se k zákazníkovi hotová aplikace dostane až na konci, kde může zjistit, že přesně neodpovídá jeho potřebám a tak se celá vyvíjí znova, až sice odpovídá původnímu zadání, ale situace se za tu dobu tak změnila, že software přesto není "aktuální" a potřebný. V případě internetových projektů, kde zákazník není přesně definován a představuje jej skupina uživatelů internetu, kteří ani nemají přesně stanovené požadavky, musí softwarová aplikace odpovídat současným trendům, jinak není potřebná. Tak se jedním z nejdůležitějších a zároveň kritických aspektů vývoje softwaru stává rychlost dodání, čímž se dostáváme ke vzniku takových metodik, které budou lépe a dynamičtěji reflektovat potřeby zákazníků a zároveň to budou dělat rychleji. Důležité rysy takových metodik je větší spolupráce a semknutí vydavatele se zákazníkem, dobrá komunikace v týmu, zpětná vazba, zredukování doby určené na analýzu a návrh, zaměření na implementační fáze a dobré otestování [11].

3.1 Agilní manifest

Nápad ke vzniku metodik, které budou lépe vyhovovat dynamicky se vyvíjejícímu světu, vznikl na počátku 21. století, kdy se v americkém Utahu roku 2001 sešlo 17 nejvýznamnějších představitelů nových přístupů k vývoji softwaru: Kent Beck, Ward Cunningham, Martin Fowler, Alistar Cockburn, Ken Schwaber, Jeff Sutherland a další, a poprvé diskutovali své návrhy s použitím slovního spojení agilní metodiky. Všichni pánové za sebou mají minulost jako špičkoví programátoři a vývojáři v několika menších i větších firmách, kde zažili mnoho neúspěšných projektů a krachů, se kterými samozřejmě nebyli spokojeni.

Před jejich setkáním předcházela delší doba práce každého z nich na vlastním návrhu nové metodiky, která by měla zamezit neúspěchům na projektech a přinést nápady a nové přístupy v programování softwarových produktů. Přestože většina pracovala odděleně, jejich návrhy se v základních nápadech příliš nelišily a tak záhy vznikla formulace Manifestu agilního vývoje (The Agile Software Development Manifesto) založená právě na společných rysech metodik všech zúčastněných, kteří společně vytvořili Alianci pro agilní vývoj softwaru (The Agile Alliance).

"*Objevujeme lepší způsoby vývoje software tím, že jej tvoříme a pomáháme při jeho tvorbě ostatním. Při této práci jsme dospěli k těmto hodnotám:*

- *Jednotlivci a interakce* *před procesy a nástroji.*
- *Fungující software* *před vyčerpávající dokumentací.*
- *Spolupráce se zákazníkem* *před vyjednáním o smlouvě.*
- *Reagování na změny* *před dodržováním plánu.*

Jakkoliv jsou body napravo hodnotné, bodů nalevo si ceníme více." [15]

3.2 Hlavní rysy

Postupně si trochu detailněji rozebereme každý ze čtyř hlavních bodů agilního manifestu, které reflektují základní rysy agilních metodik [3].

„Jednotlivci a interakce před procesy a nástroji."

Role jednotlivce se stává velmi důležitou hodnotou v celém vývojovém procesu. Klade se mnohem větší důraz na interakci mezi lidmi v týmu, jejichž role v týmu stoupají na cenně a stávají se hůře nahraditelnými. Díky komunikace vzniká mnohem více nápadů, samozřejmě kvalita komunikace hraje velkou roli. Tento bod vyjadřuje hlavně to, že je lepší mít špatně zdokumentovaný proces s dobrou komunikací mezi lidmi v týmu než mít skvěle zdokumentovaný proces a vývojový tým, který společně nekomunikuje své postřehy.

„Fungující software před vyčerpávající dokumentací."

Výsledný kód je nakonec to, co nejlépe vypovídá o tom, co tým vyvinul a co ne. O to lépe, pokud je funkční. Dokumentace obsahující rozbor požadavků, návrhy obrazovek, diagramů komunikace a další je využívána týmem k vytvoření představy, jaká může být budoucnost a spolu s jejich zkušenostmi jako hlavní body při vývoji. Je důležité umět rozhodnout, v jaké fázi je rozsah dokumentace skutečně potřebný k vyřešení problému a kdy už je jeho psaní a údržba kontraproduktivní vzhledem k jejímu použití během vývoje.

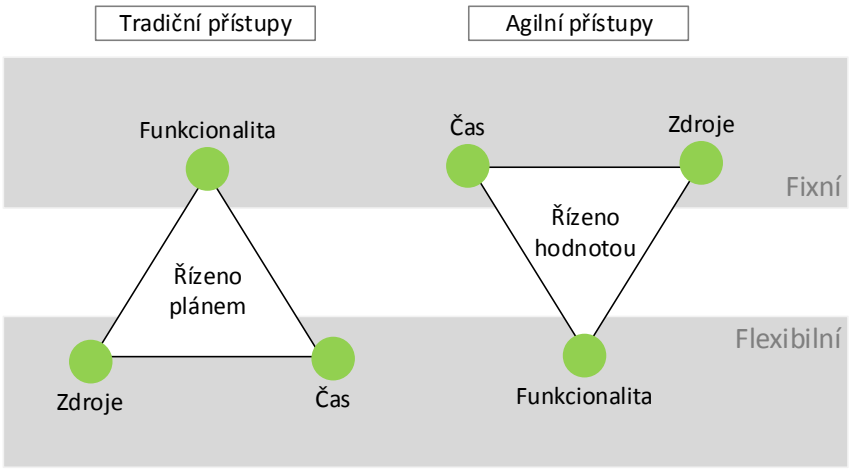
„Spolupráce se zákazníkem před vyjednáním o smlouvě."

Agilní metodiky kladou velký důraz na zapojení zákazníka do vývojového procesu. Výhodou toho je rychlejší komunikace mezi vývojovým týmem a zákazníkem a tvorba přátelštějšího vztahu, díky kterému se vzniklé problémy budou řešit snáz. Zákazník se navíc účastní společného rozhodování na věcech okolo projektu. Přibrání zákazníka do týmu mu tak dá větší motivaci k vytvoření kvalitního software, protože to je to, co obě strany chtějí nejvíce. Smlouva je čas od času užitečná, avšak dobrá spolupráce může způsobit, že jí vůbec není třeba.

„Reagování na změny před dodržováním plánu."

Vytváření plánu je užitečné pro lepší sledování prací do budoucna. Různé plánování využívají všechny agilní metodiky, každá pomocí jiných aktivit, obecně však pro všechny platí, že vytváření plánu na delší dobu dopředu není vhodné, protože tým by měl umět poměrně rychle reagovat na přicházející změny.

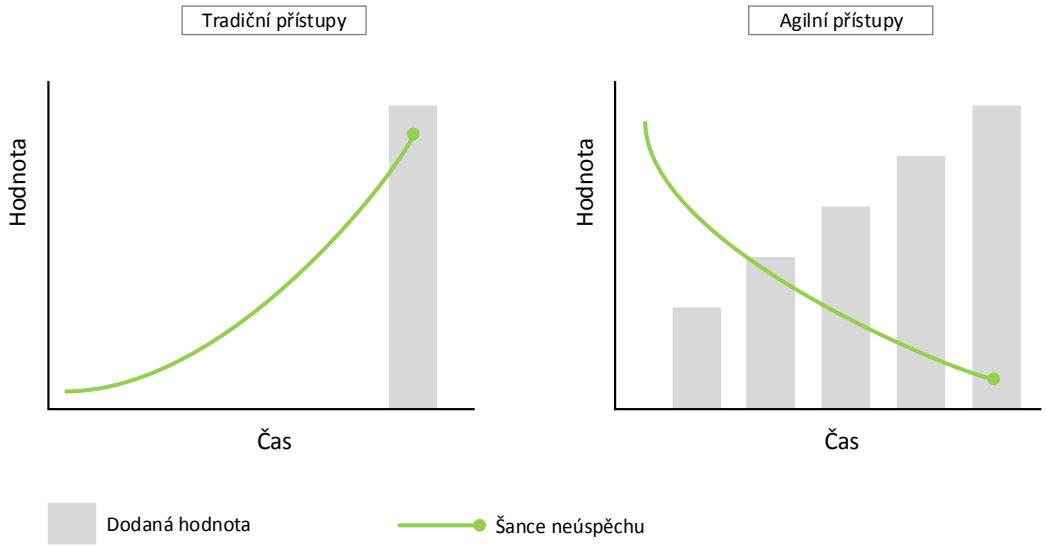
(Obrázek 1) a *(Obrázek 2)* znázorňují hlavní rozdíly agilních metodik oproti tradičním přístupům popsaných v Kapitole 2.



Obrázek 1: Rozdíly mezi tradičními a agilními přístupy z pohledu fixování času, požadavků a zdrojů

Tradiční metodiky pevně stanovují požadavky na počátku vývojového procesu, zatímco čas a zdroje jsou flexibilní s cílem splnit požadavky popsané v dokumentu Specifikace požadavků. Na začátku tedy bylo jasné, co má program umět, ale špatně se odhadovalo, za jak dlouho to bude hotové a kolik to bude stát. Oproti tomu agilní metodiky si na začátku stanoví maximální cenu a časové rozpětí pro tvorbu projektu, kterému se vývojový tým musí v průběhu řešení projektu přizpůsobit, zatímco požadavky na funkcionalitu se mohou měnit.

(Obrázek 2) ukazuje křivku zobrazující šanci pro neúspěch v porovnání tradičních a agilních metodik. U tradičních metodik na konci dostáváme plné funkční řešení − křivka neúspěchu stoupá s tím, jak v průběhu řešení nebyla možnost reagovat na změnu zadání díky malé spolupráci se zákazníkem. Oproti tomu u agilních metodik vidíme v průběhu řešení několik mezivýsledků (ne plně funkčních), díky kterým má křivka neúspěchu klesající tendenci, jak postupně vývoj reaguje na změny ze strany zákazníka.



Obrázek 2: Rozdíly mezi tradičními a agilními přístupy z pohledu dodané hodnoty za časový úsek

Množství formálních a byrokratických objektů se zmenšuje a jediným základním, nezpochybnitelným nositelem informace je zdrojový kód. Právě na jeho tvorbu je kladem největší důraz a začíná se téměř záhy po stanovení cílů projektu. Vývoj je založen na iterativním a inkrementálním principu s velmi krátkými iteracemi. Snaha o to, aby množství práce, které není nutné dělat, bylo co největší, je zásadní žádanou dovedností. Důraz se klade na co nejjednodušší pracovní postupy, které se při změně snadněji změní. Důležitá je rovněž motivace. Namotivovaný vývojový tým pracuje efektivněji než tým, který není jednotný a práce ho nebaví.

3.3 Existující agilní metodiky

Existuje několik metodik vycházejících z agilního manifestu, které jejich základní rysy trochu upravují a přidávají další pravidla. Některé z nich si v následujícím textu popíšeme:

- Extrémní programování
- Testy řízený vývoj (*„Test-Driven Development“*)
- Adaptivní vývoj softwaru
- Lean Development
- **SCRUM**

3.3.1 Extrémní programování

První z popisovaných zástupců agilních metodik je extrémní programování (dále jen XP). Její vznik se datuje na rok 1999, zakladatelem je již dříve zmíněný člen Agilního manifestu – Kent Bleck. XP je vhodné pro menší až střední vývojové týmy (hovoří se o počtu dvou až deseti programátorů), které mají k dispozici často se měnící požadavky nebo nejasné zadání ze strany zákazníka. Je to iterativní inkrementální metoda, používající běžné a známé postupy a myšlenky, které dovádí až do extrému – odtud její název, extrémní programování. V tomto textu si uvedeme hlavně ty rysy, kterými se metodika liší od základních rysů agilních metodik popsaných v předešlé kapitole [14].

XP je založena na čtyřech základních hodnotách: komunikace, jednoduchost, zpětná vazba a odvaha. Každá ze zmíněných hodnot souvisí se všemi ostatními. Podstatné informace musí být viditelné celému týmu, a proto je důležité, aby jednotlivci neměli strach ze špatných reakcí a sdíleli mezi sebou i špatné zprávy. Získávání zpětné vazby v podobě testování jak ze strany programátorů (to probíhá v řádu hodin, minut a dní) tak ze strany zákazníka (řády týdnů, měsíců) a možností tak reagovat na nalezené chyby dává projektu větší šanci na úspěšné dokončení. Protože XP považuje zdrojový kód jako jediný jednoznačný, nezpochybnitelný, změřitelný a ověřitelný zdroj informací, je na něj orientována většina pracovních postupů. Celkem dvanáct hlavních: plánovací hra, malé a časté verze, metafora, jednoduchý návrh, testování, frekventovaný refaktoring, párové programování, společné vlastnictví kódu, nepřetržitá integrace, udržitelné tempo, celý tým spolu a standardy pro psaní zdrojových kódů; jejichž bližší popis přehledně zobrazuje Příloha B *(Tabulka 1)*.

Všechny výše zmíněné postupy je nutné dodržovat, protože mezi sebou navzájem souvisí a jsou propojené. Nevýhodou metodiky je ten, že ne každý člověk se může hodit do XP týmu. Někteří programátoři odmítají nebo nemají odvahu k párovému programování a společnému vlastnictví kódu, jiným vadí permanentní komunikace.

3.3.2 Lean Development

Historicky se jeho vznik datuje do 80. let 20. století, ovlivněn výrobou aut značky Toyota v Japonsku. Jeho pojmenování do češtiny není přeloženo, avšak slovo „*Lean"* se dá přeložit jako „štíhlý" nebo „zeštíhlený", což odpovídá i popisu metodiky [11].

Důraz je kladen především na zdroje a suroviny, nástroje používané týmem při procesu vývoje, získané dovednosti a výsledný produkt dodaný zákazníkovi. Za cíl si dává zkrátit vývoj produktu na třetinu obvyklého času za třetinového rozpočtu a snížit počet výskytů chyb na třetinu. Metodika obsahuje celkem 10 doporučených pravidel a 7 principů, které pokud se budou dodržovat, vývoj bude efektivnější a tým dodá zákazníkovi výsledný produkt ekonomičtější cestou. Zmíněných 10 pravidel: odstranění zbytečností, zminimalizování zásob, z maximalizování toku, zaměření na poptávku, dát pravomoci pracovníkům, snažit se vyhovět zákazníkovi, získání zpětné vazby od zákazníka, odstranění lokálních optimalizací, vytvoření partnerství s dodavateli a nastavit kulturu pro neustálé zlepšování. Všechna pravidla spolu s jejich bližším popisem a vysvětlením zobrazuje Příloha B *(Tabulka 2).*

4 Scrum

Metodika Scrum je agilní metodika zaměřená především na zefektivnění celého vývojového procesu za použití inkrementálního a iterativního vývoje s hlavním cílem dodat fungující produkt zákazníkovi. V roce 1995 ji začali formulovat Ken Schwaber a Mike Beedle, její název je převzat z anglického slova „*scrum*“, které souvisí s americkou hrou ragby. Stejně jako se hráči na hřišti snaží dotlačit míč do cíle, tak se vývojáři v týmu snaží „dotlačit“ projekt až do cíle v podobě dodání aplikace takovou, jakou ji zákazník chtěl. Jako agilní metodiky, i Scrum vznikal proto, aby byl vývoj více flexibilní a byl schopen rychleji reagovat na měnící se prostředí a požadavky od zákazníků.

Scrum není proces pro samotný vývoj produktů, ale spíš procesní rámec v rámci kterého lze používat jiné procesy a techniky. Jeho nasazení do společností, které již nějakou dobu úspěšně používají jiné metodiky vývoje je vhodnější, protože lze pokračovat v již zažitých zvyklostech a změní se pouze způsob, jakým se vede projekt. Tři základní pilíře, na kterých popis metodiky stojí, jsou transparentnost, adaptace a kontrola. Díky transparentnosti jsou důležité části procesů viditelné všem těm členům, kteří díky nim mohou ovlivnit výsledek. Je ale důležité dodržovat pravidla pro stejný standard popisu procesů. Kontrola postupu vývoje probíhá tak často, aby tato činnost zbytečně nebrzdila skutečnou práci a nejlépe za účasti kvalifikovaného pracovníka. Díky ní je možné odhalit výkyvy od požadovaného výsledku poměrně brzy a rychle tak na ně zareagovat, s čímž souvisí třetí pilíř, adaptace [19][20].

4.1 Role v týmu

Scrumový tým obsahuje tři základní typy rolí: vývojový tým, Scrum mastera a Vlastníka produktu. Týmy jsou sebe organizující, což znamená, že si samy volí práci, kterou provedou, a měly by schopny jí dokončit bez vnější pomoci.

4.1.1 Vývojový tým

Vývojový tým je tvořen profesionály, kteří jako jediní členové scrum týmu přidávají užitnou hodnotu výslednému produktu. Každý člen týmu je zodpovědný za konkrétní množinu objektů, které vyvíjí, na rozdíl od extrémního programování, kde je zdrojový kód veřejný všem; což vede ke kvalitnějšímu zpracování kódu. Vlastník se tak více snaží o kvalitní zpracování svého kódu, protože ví, že případnou zanesenou chybu si bude muset opravit on sám. Problém nastává ve chvíli, kdy tento člověk opustí tým, protože v nejhorším případě tam nezbude nikdo další, kdo dokáže pořádně porozumět naimplementovanému kódu. Za výsledek přidané hodnoty k požadovanému produktu ale nese zodpovědnost celý tým.

Organizace vytváří vývojové týmy tak, aby si jejich členové spolu dobře rozuměli a aby jejich schopnosti a specializace pokryly rozměry žádaného produktu. Všichni jeho členové by měli společně sdílet jednu kancelář. Neexistují žádné pod týmy zabývající se konkrétní oblastí řešení projektu, všichni členové týmu tak mají titul vývojář bez ohledu na jejich konkrétní specializaci. Týmy si samostatně řídí vlastní práci, což vede ke zvýšení efektivity práce. Velikost vývojového týmu je ideálně mezi 3-9 vývojáři, obecně však platí, že tým by měl zůstat dost malý na to, aby jeho spolupráce byla stále flexibilní, ale zase ne tak malý, aby byl schopen udělat znatelnou část práce na projektu.

4.1.2 Vlastník produktu

Má na starosti definování cílů a vizí produktu, je jeho vlastníkem. S vývojovým týmem pravidelně spolupracuje a je mu v případu potřeb k dispozici. Tráví dostatek času se zákazníkem, aby pochopil cíle projektu a co, jak a proč se má udělat. Tyto informace pak přináší i zbytku týmu. Jeho hlavní činností je ovšem správa Produktového backlogu − hlavně jeho transparentnosti, a aby jeho položkám dostatečně rozuměl celý tým (viz Kapitola 4.2). Rozhoduje, které funkcionality budou vykonány dříve, či později a jednotlivým položkám přiřazuje priority tak, aby co nejlépe odpovídaly aktuálním požadavkům na výsledný projekt. Tuto roli pro konkrétní produkt zastává pouze jedna osoba.

4.1.3 Scrum master

Postava scrum mastera náleží právě jedna ke každému vývojovému týmu. Představuje jakéhosi šéfa týmu, který týmu především slouží. Stará se o to, aby byla pravidla Scrum dodržována a pomáhá s jejich osvojením jak v rámci vývojového týmu, tak v rámci celé organizace. Spolu s ostatními scrum mastery pomáhá proškolit ostatní zaměstnance společnosti a pomáhá jim určit, které činnosti směřované směrem k vývojovému týmu budou potřebné a které jej budou jen zdržovat. Snaha o minimalizaci takových činností je maximální.

Pomáhá vývojovému týmu k sebe organizaci, učí jej používat správně techniky sebe vývoje, ideálně s ním sdílí jednu kancelář. Snaží se, aby nebyl vyrušován vnějšími vlivy, které by jej mohly odvádět od soustředění na práci. Odstraňuje problémy, které týmu brání v postupu a motivuje jej k lepším výsledkům. Všechny schůzky konané během sprintů (viz Kapitola 4.3) řídí právě Scrum master. Kromě práce s vývojovým týmem má Scrum master na starost i Produktový backlog, který má za úkol udržovat transparentní pro všechny členy týmu a postarat se o to, aby byl efektivně spravován.

4.2 Backlog

Základním nositelem informací o tom, které funkce, požadavky, rozšíření a opravy chyb se mají implementovat a provést je tzv. Produktový backlog. Jeho hlavním správcem je Vlastník produktu, který do něj jako jediný přidává další položky a aktualizuje již přidané. Všem položkám obsaženým v Produktovém backlogu stanovuje prioritu, podle které bude odvozeno pořadí implementace. Každá položka tak obsahuje popis, přiřazenou prioritu a odhad trvání (tu stanovuje zásadně vývojový tým). Na základě získávání zpětné vazby od zákazníka nebo v případě změn v organizaci a požadavcích Vlastník produktu položky aktualizuje a za spolupráce vývojového týmu zpřesňuje odhad doby implementace a stanovené priority. Položky mohou být upravovány kdykoliv za účasti Vlastníka produktu. Mělo by platit, že položky, které jsou v seznamu výše (tedy mají větší prioritu), jsou popsány s větší mírou detailu než ty, které jsou položeny níže. Pro jeden produkt existuje právě jeden Produktový backlog, pokud na vývoji produktu pracuje zároveň více vývojových týmů, položky si rozdělují.

Důležitým atributem každé z položek Produktového backlogu je tzv. *„Definition of done“* (Definice toho, co je hotovo; často se používá pouze zkratka DoD). Ta jasně definuje, co všechno musí být splněno, aby bylo možné danou položku, resp. přírůstek produktu, považovat za funkční a hotovou. Na základě tohoto popisu se tým může rozhodovat, které všechny položky příjme do sprintu. Kromě DoD pro jednotlivé položky existuje i DoD obecné pro všechny položky. Pokud na produktu spolupracuje více týmů, je nutné definice sjednotit.

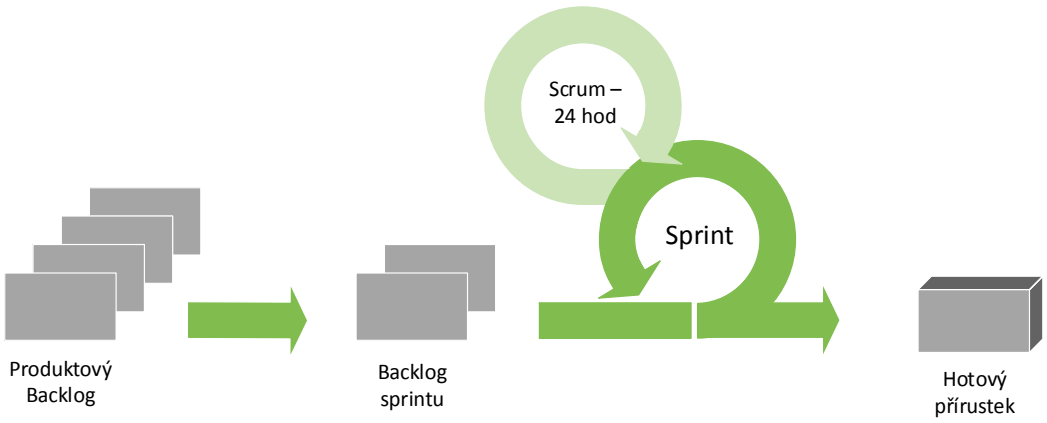
Během doby plánování sprintu (viz Kapitola 4.3) celý scrum tým společně vybere položky s nejvyšší prioritou (s ohledem na kapacitu vývojového týmu), které přesune do tzv. Backlogu sprintu. Každý tým

má svůj Backlog sprintu. Ten obsahuje všechnu práci, kterou si daný tým dal za cíl za jeden sprint udělat. Vývojový tým jej upravuje v průběhu celého sprintu (je čistě v jeho režii), jak pokračují práce a objevují se nové detaily a zpřesňují se odhady. Aktuální stav Backlogu sprintu okamžitě vypovídá o tom, na čem zrovna daný tým pracuje.

4.3 Sprinty

Celý vývojový proces je rozdělen do kratších iterací, které se nazývají sprinty. Délka jednoho sprintu je obvykle jeden měsíc, avšak může být přizpůsobena potřebám organizace. Důležité však je, aby všechny sprinty trvaly stejně dlouho. Ihned po dokončení jednoho sprintu následuje další. Cílem sprintů je dodat funkční nasaditelný přídavek do výsledného produktu. Jejich součástí jsou řídící činnosti, např. řízení rizik, které zabraňují vzniku chaosu, přesto ale vývoj produktu musí zůstat flexibilní.

Celkový průběh vývoje metodikou, včetně zařazení sprintů do vývojového procesu zobrazuje *(Obrázek 3)*.



Obrázek 3: Průběh metodiky Scrum

Během sprintu se nesmí provádět takové změny, které by mohly ohrozit cíle sprintu. Pokud je délka sprintu příliš dlouhá a v době práce na aktuální iteraci se zásadně změní podmínky v organizaci, technologické podmínky nebo požadavky na cílovou funkcionalitu, má Vlastník produktu výlučnou možnost sprint před jeho ukončením zrušit. Po jeho zrušení se vyhodnotí již naimplementované položky, které zákazník většinou akceptuje. Zbylé položky se vrací do Produktového backlogu. Rušení sprintů je spíše výjimečnou situací, pokud k ní dojde, může to na tým mít velmi špatný dopad.

4.4 Schůzky

V průběhu sprintu tým plánuje několik schůzek, které mu pomáhají při vývoji produktu. Všechny schůzky přímo podporují jednu z hlavních rysů metodiky Scrum – komunikaci, a tedy by se měly dodržovat.

4.4.1 Plánovací schůzka

Plánovací schůzky (*„Sprint Planning“*) se účastní celý vývojový tým, Vlastník produktu i Scrum master. Ten má na starost její celé vedení a snaží se, aby její konání nepřesáhlo délky osmi hodin (pro sprinty kratší než jeden měsíc je plánovací úměrně schůzka kratší). Cílem této schůzky je naplánovat práci, která bude vykonaná v rámci příštího sprintu. Důležité informace, které tým pro plánování práce potřebuje vědět, jsou: dostupná kapacita vývojového týmu pro další sprint, podaný výkon v předešlém sprintu, vytvořený přírůstek z předešlého sprintu a seznam položek Produktového backlogu. Po naplánování práce tým definuje cíl celého sprintu. Ten pomáhá udržovat konzistenci a jednotnost celého týmu.

V rámci plánovací schůzky tým začíná uvažovat nad tím, jak danou práci udělá a začíná dělat návrh systému a prací nutných k přeměně položek na požadovaný přírůstek. Na první dny nového sprintu je práce rozdělena na menší části a naplánována na dny. Na jejím konci by měl mít vývojový tým jasno, jakým způsobem hodlá dosáhnout cíle sprintu a vytvořit tak přírůstek produktu.

4.4.2 Denní schůzka

Denní schůzka (*„Daily scrum“*) je 15-ti minutová schůzka (z názvu jasně vyplývá, že se koná každý pracovní den), které se účastní vývojový tým a Scrum master. Jejím cílem je, aby se členové vzájemně seznámili s tím, na čem zrovna pracují, co za uplynulý den udělali a co mají v plánu udělat zítra. Diskutují se možné hrozby, které týmu mohou zabránit splnit cíl sprintu v ohraničením čase. Scrum master dbá na to, aby se schůzky účastnili pouze členové vývojového týmu, moderuje diskuzi a dává pozor, aby nebyl přesáhnut její časový limit.

Schůzka zajišťuje dobrou komunikaci v týmu, znalost celého týmu o projektu a práci ostatních, povědomí o možných překážkách a jejích řešení a může eliminovat nutnost dalších schůzek, které by jen zbytečně zdržovaly samotnou implementaci. Podporuje možnost tvorby rychlých rozhodnutí (podpora adaptace). Navíc díky této schůzce může tým odhalit své slabší články, protože se velmi snadno ukáže, kdo co dělá a kdo co dlouhodobě nedělá.

4.4.3 Vyhodnocení sprintu

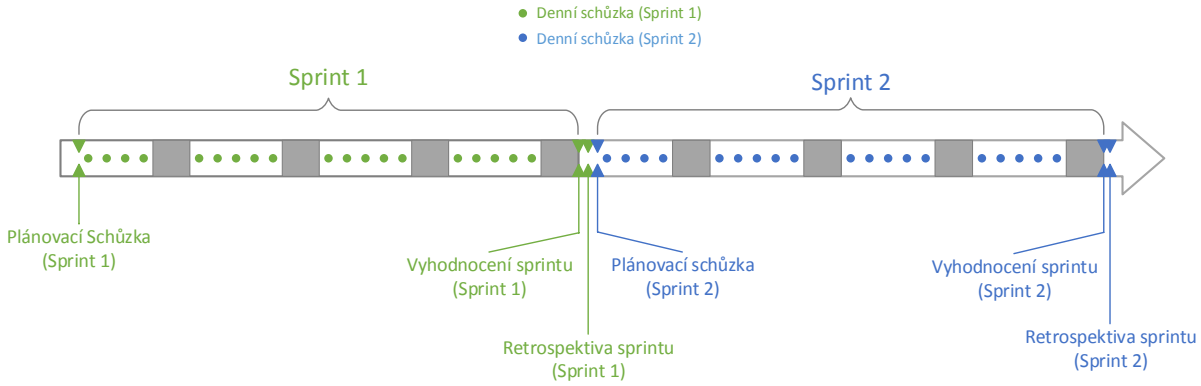
Vyhodnocení sprintu (*„Sprint Review“*) se provádí na konci každého sprintu a účastní se jí Scrum master, vývojový tým, Vlastník produktu a všichni ostatní, kteří se podíleli na vývoji přírůstku a byli pozváni Vlastníkem produktu. Pro sprint dlouhý jeden měsíc, by délka schůzky neměla přesáhnout čtyři hodiny. Cílem setkání je společně probrat hotový přírůstek za poslední sprint, zhodnocení co se povedlo, kde se narazilo na chyby, jak se s nimi tým. Na základě těchto informací se aktualizují položky Produktového backlogu a zpřesní se odhady na dokončení celého produktu. Zároveň tým může lehce nastínit, které z položek pravděpodobně budou přiřazeny do dalšího sprintu.

4.4.4 Retrospektiva sprintu

Retrospektiva sprintu probíhá na konci každého sprintu a účastní se jí vývojový tým spolu se Scrum masterem, který opět celou schůzku moderuje. Pro jedno-měsíční sprint by neměla přesáhnout délku tří hodin. Cílem tohoto setkání je zaměřit se na poslední sprint ve spojitosti s použitými procesy, nástroji a komunikaci mezi lidmi v týmu. Tým diskutuje aspekty vývoje, které probíhají v pořádku, a hledá takové, které by se daly vylepšit. Výstupem schůzky by měl být seznam všech vylepšení, které by

v příštím sprintu mohly zajistit efektivnější spolupráci týmu a lepší průběh sprintu. Vylepšení se během příštího sprintu zavedou a na další retrospektivě tým může diskutovat, jestli byla užitečná, či nikoliv.

(Obrázek 4) zobrazuje časovou osu, na které jsou přehledně znázorněny všechny schůzky během sprintů. Mohlo by se zdát, že je jich zbytečně moc a tým zdržují od samotného vývoje, ale praxe ukazuje, že to není pravda. Investice do schůzek se vyplácí, protože nutí členy týmu mezi sebou komunikovat a bavit se o problémech ve vývoji, což v budoucnu může zajistit menší počet chyb a rychlejší vývoj produktu.



Obrázek 4: Časová osa schůzek během sprintu

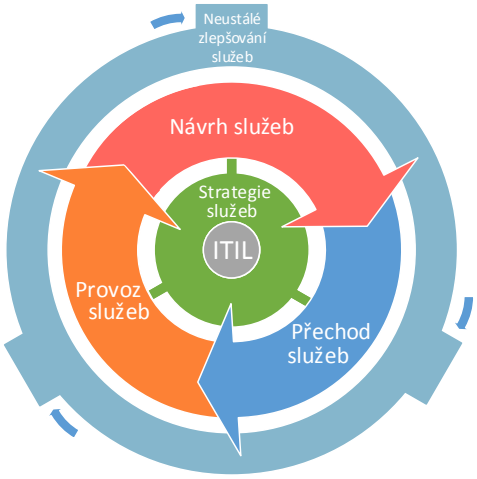
4.5 Závěr

Metodika Scrum je extrémně jednoduchá co se týče pravidel, ovšem její správné zavedení do organizace, kde se již používá jiná metodika, může být naopak extrémně těžké. Ne všechny typy lidí jsou vhodné pro tento druh vývoje a zavádění nových pravidel, proto bude třeba nalézt ty, kterým nový způsob práce bude vyhovovat a vývojový tým tak podle potřeby obměnit. Způsob, jakým nejlépe Scrum zavést a používat v organizaci, kde se již několik let používá ITIL se budeme zabývat zbytek této práce.

5 ITIL

„Information Technology Infrastructure Library“ (Knihovna Infrastruktury Informačních Technologií), dále jen ITIL, je soubor navrhovaných konceptů a postupů, poskytující sadu procesů, které jsou spolehlivé, efektivní a přizpůsobivé společnostem všech velikostí. Účelem navrhovaných postupů je pomoci společnostem lépe naplánovat a využívat zdroje informačních technologií, což vede i ke zlepšení poskytování vlastních služeb zákazníkům. ITIL, jakožto procesní rámec, je nutné implementovat, jelikož obsahuje pouze obecná doporučení a nezabývá se konkrétními situacemi.

ITIL vznikl v 80. letech 20. století ve Velké Británii pod záštitou CCTA (*„Central Communications and Telecommunications Agency“*) za účelem vylepšit úroveň kvality IT služeb. Jeho první verze obsahovala 46 - 50 souvisejících knih a ve svých počátcích byl využíván především ve Velké Británii a Nizozemí. V letech 2000 až 2004 vydává OGC (Office of Government Commerce), druhou verzi ITIL jako soubor zrevidovaných osmi knih. Třetí verze OGC vydává v roce 2007 jako soubor pěti knih pokrývající životní cyklus služeb IT společně s oficiálním úvodem. V roce 2011 dochází zatím k poslední aktualizaci ITILv3 na ITIL 2011.



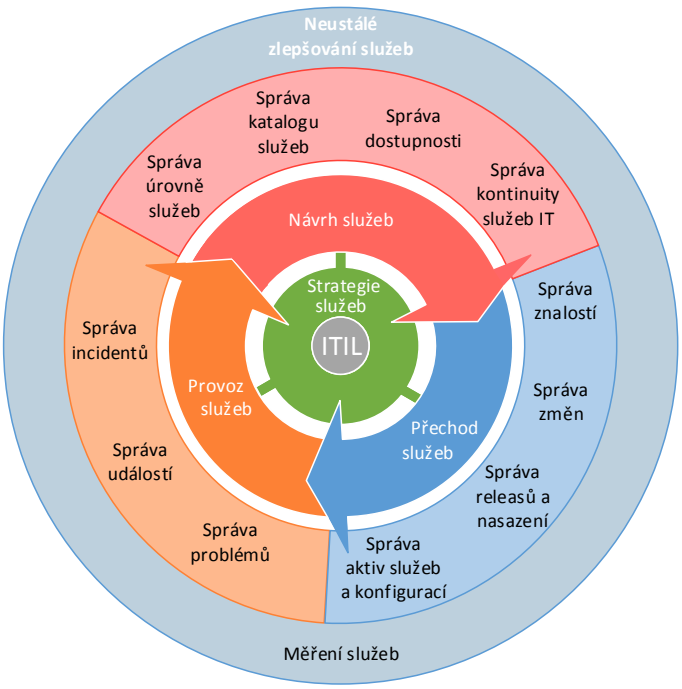
Obrázek 5: Fáze ITIL

Rámec ITIL je založen na pěti fázích životního cyklu služeb *(Obrázek 5)*, které v podstatě pokrývají celou dobu životnosti služby. Každá z fází cyklu je popsaná v jedné z pěti knih zajišťující nejlepší vedení a praktiky pro danou fázi. Od analýzy požadavků businessu ve Strategii služeb *(„Service Strategy“)*, skrz Návrh služeb (*„Service Design“*), Přechod služeb (*„Service Transition“*) a Provoz služeb (*„Service Operation“*), až k udržení kvality poskytování IT služeb v Neustálém zlepšování služeb (*„Continual Service Improvement“*). Každá fáze cyklu má vliv na další fáze a její vstupy jsou závislé na výstupech ostatních fází, což zajišťuje rychlé přizpůsobení služeb v závislosti na změnách poptávky na trhu s poskytováním služeb.

Než si postupně rozebereme jednotlivé fáze životního cyklu služby, zaměříme se na pojem služba. Jak již bylo v úvodním přehledu pojmů řečeno, služba je prostředek poskytování hodnoty zákazníkovi, kdy zákazník nemusí nést přímé náklady a rizika se zavedením služeb. Služba IT je speciální případ služby, založené na použití informačních technologií. ITIL má za hlavní cíl podporu zákazníka pomocí poskytování služeb a správou služeb IT. Správa služeb je systém specifických organizačních schopností (zahrnující procesy, metody, funkce, role a činnosti) pro dodávání hodnoty zákazníkovi ve formě

služeb. Vstupem do správy služeb jsou zdroje a způsobilosti, reprezentující aktiva poskytovatele služeb. Ty jsou za pomocí procesů a metod transformovány na kvalitní služby dodávané zákazníkovi.

Jednotlivé knihy popisující fáze životního cyklu služby jsou velmi obsáhlé a do detailu popisují všechny důležité procesy, postupy, dokumenty a role důležité pro jejich implementaci v organizaci. Pro potřeby této práce bude v následujících kapitolách každá z fází popsána, oproti originálu, velmi stručně, a kromě hlavních cílů a zaměření si popíšeme hlavně ty procesy, které budou hrát podstatnou roli v praktické části této práce. Všechny tyto procesy můžete vidět graficky rozdělené do dané fáze *(Obrázek 6)*. Hlavním zdrojem, který mě při psaní celé kapitoly o ITIL provázel, je kniha „ITIL 2011“ vydavatelství Computer Press [1], doplněná o originální knihy o ITIL 2011 [6] [7][8][9][10] a [2].



Obrázek 6: Fáze ITIL spolu s jejich důležitými procesy

5.1 Strategie služeb

První z fází (knih), Strategie služeb (*„Service Strategy“*), se zabývá zevrubným pochopením zákazníkových potřeb, definicí návrhu strategií a naplánování postupu, kterým chce organizace dosáhnout vytyčených cílů. Hlavním cílem této fáze je přeměnit správu služeb na strategické aktivum. Jako jeden z klíčových konceptů definuje strategii 4P:

- perspektiva – vize a směr organizace,
- pozici – pozice organizace na trhu z pohledu zákazníka,
- plán – jak poskytovatel dosáhne stanovených cílů,
- profil – charakteristické znaky při rozhodování a chování v průběhu doby.

Poskytovatelé služeb jsou předmětem konkurenčních sil na trhu, existuje několik druhů od interních (poskytovatel a zákazník v jedné firmě), sdílený (jeden poskytovatel pro více částí firmy) a externí (poskytovatel a zákazník každý z jiné firmy). Pro poskytovatele je důležité definovat trh a zákazníky, pochopit zákazníka, na základě výsledků definovat, klasifikovat a vizualizovat služby, navrhnout balíčky služeb atd. Zákazníka ve výběru poskytovatele služeb ovlivňují aspekty, jako jsou transakční

náklady nebo výhody spojené s jeho výběrem oproti konkurenci. Hodnota poskytované služby z pohledu zákazníka je popsána pomocí dvou důležitých komponent: užitečnost služby a její záruka. Užitečnost vyjadřuje přidanou hodnotu, kterou zákazník dostává ve smyslu podporovaných výstupů nebo odstraněných omezení a záruka je spojena s podporou bezpečnosti, dostupnosti, kapacity a kontinuity služby.

Důležitou otázkou v rámci firmy je rozhodnout, které úkony IT bude firma řešit sama a které „nakoupí". Na základě zhodnocení komplexnosti a výkonnosti při interním zpracování služeb se rozhodne, zda firma bude hledat plné externí řešení, selektované externí řešení nebo zůstane u interních zdrojů. Klíčové procesy strategie služeb se zabývají finančním základem pro fungování firmy a manažerským řízením správy služeb, jsou zaměřený buďto na interního nebo externího zákazníka.

Klíčovou rolí ve fázi Strategie služeb je Produktový manažer (*„Product Manager"*, PM), který nese zodpovědnost za rozvoj a správu služeb během celého jejich životního cyklu. Má zodpovědnost za vhodné zajištění kapacit pro vyvíjenou službu a plánování vývoje služby včetně všech souvisejících náležitostí.

5.1.1 Správa strategie služeb

Hlavní proces Správy strategie služeb generuje strategické cíle v souladu s podnikovou strategií společnosti. Vstupem procesu jsou požadavky nebo existující plány strategie a výsledky jednání s potencionálními nebo stávajícími dodavateli. Vlastníkem procesu je člen vrcholového vedení, který řídí činnost ostatních vlastníků procesů tak, aby došlo k začlenění strategie do všech ostatních procesů. Mezi aktivity procesu patří strategická analýza, v rámci které jsou analyzovány zdroje a dovednosti organizace, jsou posuzovány interní a externí podmínky mající vliv na poskytované služby a odhalují se slabé a silné stránky organizace analýzou konkurence. Další, generování strategie, má za cíl vymezit místo na trhu, kde chce organizace úspěšně oslovit zákazníky, a udržet své strategické výhody vůči konkurenci. Následuje fáze dodržování strategie po dobu životního cyklu služby, kdy se zvolená strategie zoptimalizuje přidáním dalších procesů, pro docílení maximálního uspokojení zákazníkových potřeb. Nakonec měření a vyhodnocování výsledků napomáhá pro budoucí zlepšování dodávaných služeb a expanze společnosti. Výstupy procesu jsou nové strategické plány a strategické požadavky na nové služby. Rozhodujícími faktory úspěchu jsou dobrá znalost externích a interních trendů a skutečné aktuální informace o podniku. Ukazatele výkonnosti mohou být např. výkyvy od naplánované strategie a dosažení cílů.

5.1.2 Správa financí služeb IT

Tento proces má za hlavní cíl finanční zhodnocení možnosti vytvoření nové služby nebo správy služeb, správu rozpočetnictví, provozní účetnictví a zpoplatnění, výpočet reálných nákladů a rozdělené nákladů zaměřených pro fungování služby a pro zákazníka. Businessu dodává vypočítanou hodnotu služeb a ohodnocení provozního fungování. Vstupem procesu je portfolio služeb, předpisy a normy organizace nebo výstup zpráv auditu týkající se rozpočtování, účtování nebo zpoplatnění. Vlastník procesu má za úkol přípravu rozpočtu. Mezi nejdůležitější aktivity procesu patří účtování, v rámci kterého se rozdělují náklady dle přidělení a variability; a kontrola, jestli tyto náklady pokryty z rozpočtu a výnosů. Rozpočtování se zabývá tvorbou rozpočtů za pomocí předpovídání nákladů a správou z plánovaných výdajů s ohledem na předchozí hospodářské cykly. V rámci zpoplatnění se vytvářejí cenové předpisy pro nabízené služby, vyčíslují se zúčtovatelné úkony a vytváří se cenové modely pro účtování zákazníka. Výstupem procesu jsou výsledky analýz dopadu na business, investiční analýzy, zoptimalizované náklady a další.

5.1.3 Správa portfolia služeb

Tento proces popisuje a dokumentuje všechny služby, které poskytovatel služeb navrhl, vyvinul, nabízí nebo nabízel včetně těch, které jsou teprve ve fázi konceptu. Tento proces souvisí s celkovým plánováním portfolia služeb a s přidělováním zdrojů. Z portfolia služeb se vytváří nabídka poskytovaných služeb, která by se měla shodovat s poptávkou, jinak hrozí obchodní neúspěch poskytovatele. Portfolio služeb sestává ze zásobníku, který obsahuje databáze všech plánovaných a vyvíjených služeb pro možné budoucí zavedení do nabídky; a z katalogu služeb, který obsahuje všechny nabízené služby. Dále existují vyřazené služby, které zákazníkům nejsou dále nabízeny. Vstupem procesu je změna poptávky na trhu, plány strategií nebo stížnosti od zákazníků. Vlastník procesu je zodpovědný za udržování portfolia služeb. Mezi důležité aktivity procesu patří definování katalogových služeb, analýza, schválení a záměr s přiřazením zdrojů konkrétnímu projektu. Všechny tyto aktivity probíhají v cyklu. Výstupem procesu je samotné portfolio služeb.

5.1.4 Správa požadavků

Proces Správa požadavků se snaží předvídat a řídit požadavky od zákazníků, zajišťuje pro ně vhodné kapacity pomocí procesu Správa kapacit. Chování zákazníků se odráží ve „vzoru chování v rámci obchodní činnosti“ (*„Patterns of Business Activity“*, PBA), který obsahuje předvídatelné a opakující se události v obchodním prostředí (sezónní výprodeje), které ovlivňují zákazníkovy potřeby. Vstupem procesu jsou požadavky od zákazníka, poptávka, portfolio služeb nebo obchodní analýzy. Hlavními aktivitami procesu jsou identifikace poptávky a jejich příčin, identifikace PBA, propojení uživatelských profilů s rolemi, snaha o minimalizace nákladů pro zavedené služby, vývoj diferencované nabídky služeb a další. Výstupem procesu jsou PBA a profily zákazníků. Proces se opakuje ve všech fázích životního cyklu vývoje služby.

Dalšími procesy Strategie služeb jsou Správa vztahů s businessem a Správa rizik.

5.2 Návrh služeb

Druhá fáze životního cyklu služby (a druhá kniha ITIL), Návrh služeb (*„Service Design“*), se zabývá návrhem a tvorbou služby v souladu se strategií služeb, včetně návrhu její architektury, procesů, dokumentace a politiky tak, aby bylo dosaženo požadavků businessu. Během této fáze jsou postupně zpracovávány metriky a měřící metody, které poskytují vstup pro fázi neustálého zlepšování služby, a identifikují se a spravují se rizika. Pro správný návrh služeb je důležité efektivně a hospodárně nasadit koncept 4P do praxe:

- Personál – lidé a jejich dovednosti potřebné pro provozování IT služeb,
- Procesy – aktivity a s nimi spojené role týkající se zajištění IT služeb,
- Produkty – služby, nástroje a technologie užívané pro dodávku IT služeb,
- Partneři – dodavatelé a obchodníci potřební pro podporu poskytování IT služeb.

Při návrhu služby se sbírají požadavky, dbá se na jejich komplexní zdokumentování a odsouhlasení, následný návrh služby včetně používaných technologií a procesů, navržení metod pro zjištění splnění zákazníkových potřeb, kontrola navržených procesů a jejich případné přepracování, udržování vývojové dokumentace, zhodnocení rizik a zajištění, aby se výsledky plánování shodovaly se strategií podniku. Životní cyklus návrhu služby zobrazuje Příloha A *(Obrázek 29)*.

Výstupem této fáze je balíček návrhu služby (*„Service Design Package“*, SDP), který obsahuje zdokumentované všechny aspekty určité IT služby a její požadavky pro každou fázi jejího životního cyklu. Pro každou novou, změněnou existující nebo vyřazenou IT službu obsahuje její definici, strukturu, specifikaci návrhu a rozhraní.

5.2.1 Správa katalogu služeb

Katalog služeb obsahuje informace o všech provozovaných IT službách, poskytuje businessu přesný obraz všech dostupných služeb a jejich stavu. Cílem procesu je zajistit, aby informace v katalogu služeb byly korektní a aktuální. Katalog je uvnitř hierarchicky rozdělen na služby směřované k zákazníkovi nebo podpůrné služby, a z pohledu cílové supiny jej lze ještě rozdělit na Obchodní katalog služeb (tvoří vazbu mezi obchodními procesy a služby) a Technický katalog služeb (obsahuje detaily ke všem službám). Vstupem procesu jsou obchodní informace a požadavky na portfolio služeb. Vlastník procesu je zodpovědný za vytvoření a udržování aktuálnosti katalogu služeb. Výstupem je kromě aktuálního obsahu katalogu i dokumentace schválených definic služeb.

5.2.2 Správa úrovně služeb

Tento proces popisuje úrovně dodávaných služeb IT zákazníkům, průběžně monitoruje kvalitu a kvantitu dodávaných služeb a kontroluje, jestli úrovně dodávaných služeb splňují obchodní požadavky organizace. Důležitým dokumentem je tzv. Dohoda o úrovni služeb (*„Service Level Agreement“*, SLA), která popisuje netechnicky psanou dohodu mezi zákazníkem a poskytovatelem o dodávaných IT službách a slouží jako smlouva o poskytování a řízení. Dohoda o úrovni provozních služeb *(„Operation Level Agreement“*, OLA), smlouva mezi dodavatelem a interním úsekem téže organizace, popisující podpůrné interní služby pro zajištění dodání poskytovaných služeb. Podpůrná smlouva (*„Underpinning Contract“*, UC) je smlouva mezi dodavatelem a externím dodavatelem podpůrných služeb a popisuje podmínky dodávání služby z určité oblasti.

Vstupem procesu jsou právě SLA, OLA, UC a další požadavky. Vlastník procesu je zodpovědný za správné identifikování požadavků ze strany zákazníka spojené s dodáváním služby a uzavírá s ním dohody. Výstupem jsou zprávy o službách a přepracované, popř. nové smlouvy.

5.2.3 Správa dostupnosti

Dostupnost představuje schopnost služby nebo komponenty uživateli dát funkční řešení v daný okamžik nebo v průběhu časového úseku. Definováním dostupnosti se zaručí, že dodávaná služba je pro zákazníka užitečná. Proces Správa dostupnosti se zabývá nejenom definicí dostupnosti pro poskytované služby, ale i měřením a udržováním, plánováním a implementací dostupnosti, určuje role a navrhuje procesy pro udržení dostupnosti. Zajišťuje, že jsou cíle dostupnosti měřeny a dosahovány, a že jsou dohodnuty v souladu s obchodními potřebami. Výpočet dostupnosti:

$$Dostupnost\ [\%] = \frac{Doba\ provozu\ služby - Celková\ doba\ výpadků}{Doba\ provozu\ služby} * 100 \qquad (1)$$

Jedním z aspektů dostupnosti je i spolehlivost, která ukazuje, kdy a na jak dlouho je služba bez problémů dostupná pro uživatele; a udržovatelnost, která ukazuje, jak rychle je možné po výpadku efektivně obnovit funkčnost dané služby. Servisovatelnost ukazuje schopnost externího dodavatele podpůrných služeb plnit závazky vůči organizaci popsané ve smluvních specifikacích. Vstupem

procesu jsou obchodní informace, smluvní cíle služeb nebo informace z portfolia služeb. Aktivity v rámci procesu se dělí na reaktivní (do nichž patří monitorování, měření, analýza, správa událostí a incidentů) a proaktivní (plánování, návrh, správa rizik, doporučení a zlepšování dostupnosti). Výstupem může být plán dostupnosti nebo návrhy pro možná vylepšení.

5.2.4 Správa kontinuity služeb IT

Organizace využívají pro poskytování služeb zákazníkům notnou podporu informačních technologií a tak je velmi důležité, aby při selhání nebo mimořádné situaci systém zůstal funkční a dále poskytoval zákazníkům kvalitní služby. Proces Správa kontinuity služeb IT se zabývá analýzou možných rizik, návrhu preventivních opatření k předejití kritických situací, plánováním obnovy dodávání služeb po nastalé kritické situaci nebo výpadku a rychlý návrat na úroveň poskytování služeb popsanou v SLA. Plány pro obnovu služeb IT jsou průběžně udržovány v souladu s prioritami a plány kontinuity businessu pravidelným prováděním analýz dopadů na business (*„Business Impact Analysis“*, BIA). Vstupem procesu jsou strategické informace organizace, výsledky testů kontinuity služeb nebo výsledky správy změn, a výstupem jsou schválené scénáře, plány, pravidla pro udržení kontinuity a výsledky ze správy rizik a BIA.

5.3 Přechod služeb

Třetí kniha ITIL a třetí fáze přechodu služeb, Přechod služeb (*„Service Transition“*), tvoří spolu s fázemi návrh služby a provoz služby vnitřní životní cyklus služby. Její rolí je převod strategií služby popsaných v balíčku návrhu služby do provozního využití. K tomuto jsou nutné všechny součásti potřebné pro souvislý provoz služby. Cílem fáze přechodu služeb je vytvořit prostředí mezi vývojem a provozem služby, vyvíjené změny služeb naplánovat, řídit a provádět, s čímž se pojí plánování kapacit zdrojů pro vývoj, zabalení, testování a nasazení releasů v souladu s požadavky zákazníků a obchodními požadavky. Pojem release můžeme chápat jako jednu nebo více změn IT služeb, které jsou sestaveny, otestovány a nasazeny všechny najednou, s tím, že jediný release může obsahovat najednou změny jak softwaru nebo hardwaru, tak i změny dokumentace, procesů nebo komponent.

Klíčové principy zahrnují používání obecných rámců a standardů pro opakovatelné pozdější použití, vytváření a udržování vztahů se všemi zainteresovanými stranami, zavedení řízení pro celý životní cyklus služby, porozumění všem službám a jejich užitečnosti, plánování releasů a řízení změny směrování, správa zdrojů a zlepšování kvality během přechodu služby.

5.3.1 Správa změn

Změna služby je přidání, odstranění nebo modifikace služby nebo její komponenty spolu s její dokumentací. Proces spravuje zaznamenání a vyhodnocení všech změn, přiřazení jim priority, zajišťuje jejich otestování, implementaci a zdokumentování. Změny by měly probíhat rychle a kontrolovaně používáním standardizovaných metod a postupů a celkové riziko businessu tak bylo optimalizované. Žádost o provedení změny je zadávána skrze Žádost o změnu (*„Request of Change“*, RfC) pro jednu nebo více konfiguračních položek. Životní cyklus změny dokumentuje tzv. Záznam o změně (*„Change Record“*), který obsahuje všechny údaje o změně a je uložený v systému správy konfigurací (CMS). Typy změn můžeme rozdělit na normální, standardní (předschválené s nízkým rizikem) a urgentní (potřeba rychlého zpracování). Vstupem procesu jsou jednotlivé požadavky na změny, pravidla pro správu změn a releasů. Výstupem jsou provedené nebo odmítnuté požadavky na změny nebo aktualizovaný plán změn. Aktivity v rámci procesu jsou zadávání a evidování požadavku na změnu,

její posouzení, rozbor a vyhodnocení, schválení sestavení změny, koordinace sestavení a testování, nasazení změny a nakonec uzavření záznamu o změně.

5.3.2 Správa znalostí

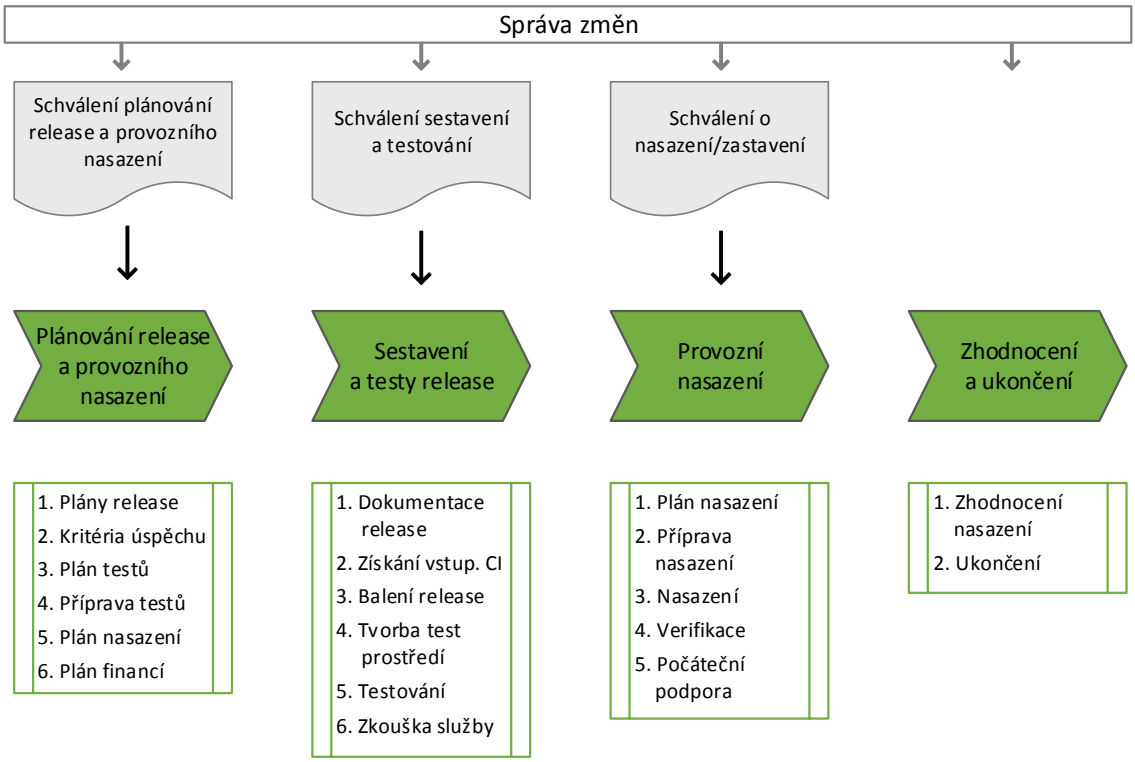
Cílem procesu Správa znalostí je zajistit pro důležité lidi aktuální, správné, bezpečné a přesné informace v potřebný čas a kvalitě pro učinění správných rozhodnutí. Takto zajištěný tok informací podporuje zkvalitnění rozhodovacího procesu, lepší vyhodnocování situací a kontextů, a snížení rizik. Proces udržuje Systém správy znalostí o službách (*„Service Knowledge Management System“*, SKMS) obsahující a poskytující potřebné znalosti a informace pro danou cílovou skupinu organizace. Proces je aktivní v rámci celého životního cyklu vývoje služby a v celém podniku. Týmy poskytují informace dalším týmům, a proto je vhodné popisy procesů a postupů pravidelně revidovat, popř. přepisovat. Správa znalostí je postavena na modelu data → informace → znalost → moudrost, který přeměňuje nevyužitá data do hodnotných informací a znalostí. Vstupem procesu jsou data, informace a výstupem aktualizovaný SKMS.

5.3.3 Správa aktiv služeb a konfigurací

Cílem procesu je udržovat aktuální, správné a úplné informace o všech aktivech (zdroje podílející se na poskytování služeb) a vztazích mezi nimi, které vytváří infrastrukturu organizace. Aktiva a konfigurační jednotky jsou nejprve identifikovány a specifikovány, kontrolovány a řízeny a jsou o nich pravidelně podávány zprávy pro zajištění integrity během celého životního cyklu služby. Správa komplexních služeb a infrastruktur IT potřebuje podporu systému CMS, který obsahuje informace o konfiguračních jednotkách a udržuje vztahy mezi komponentami služeb. Vstupem procesu jsou konfigurační informace, výsledek auditu a obsahy balíčků služeb, výstupem jsou pak aktualizované konfigurační záznamy nebo zprávy o stavu.

5.3.4 Správa releasů a provozního nasazení

Cílem procesu Správa releasů a provozního nasazení je plánování, sestavení, otestování a nasazení release do produkce, což přímo dodává zákazníkovi hodnotu, kterou požadoval a zaplatil. Proces se stará, aby všechny součásti pro vytvoření balíčku release byly všemi důležitými osobami schváleny, jejich komponenty byly vzájemně kompatibilní a jeho nasazení do produkce nezpůsobilo kolizi s již fungující poskytovanou službou. Zajišťuje i samotné vytvoření balíčku včetně zajištění pro to potřebných zdrojů. Před nasazením do produkce je vhodné vydávaný balíček otestovat nejprve na speciálním testovacím prostředí a zjištěním, zda nová nebo upravená služba tak bude stačit požadavkům na funkčnost a spolehlivost (samotné testování spravuje proces Validace a testování služby jakožto součást fáze Přechodu služby) teprve pak zajistit všechny potřebné kroky pro jeho nasazení na produkční prostředí. Pokud po nasazení testování a validace vrátí pozitivní výsledek, přechod se vyhodnotí a uzavře. Všechny tyto činnosti ideálně probíhají optimálně rychle a efektivně s přiměřeným použitím zdrojů, což snižuje rizika a přináší maximální hodnotu pro business. Posloupnost jednotlivých aktivit zobrazuje *(Obrázek 7)*. Vstupem procesu jsou schválené změny a modely releasů a výstupem kromě upravené služby jsou i změny v SLA, OLA a plánech releasů.



Obrázek 7: Posloupnost činností procesu Správa releasů a provozního nasazení

5.4 Provoz služeb

Další z fází životního cyklu služby a čtvrtá kniha knihovny ITIL, Provoz služeb (*„Service Operation“*), se zabývá kontrolou a řízením technologií, infrastruktury a aplikací, které podporují dodání kvalitních služeb zákazníkům a uživatelům. Kromě řízení provádění dříve navržených služeb, se tato fáze zabývá i sledováním nákladů na službu vzhledem k dodané kvalitě, udržováním stabilního ale i flexibilního vývoje, zajištění dobrých reakcí na poruchy a zároveň proaktivní provádění vylepšení. Hlavní je dodržet úrovně kvality dodávané služby popsané v SLA a zajistit bezchybný provoz služeb.

5.4.1 Správa událostí

Událost je změna stavu, která je důležitá pro správu služby IT nebo konfigurační jednotky. Obvykle nastalá událost může indikovat, že nějaká část nepracuje správně, což vede k zadání incidentu. Kromě problému ale může jít pouze o rutinní aktivitu nebo potřebu normálního zásahu. Proces Správa událostí má na starost správu všech aktivit týkajících se detekování, řízení a správu událostí během jejich celého životního cyklu. Po vzniku jsou události detekovány konfiguračními jednotkami nebo nástroji správy a dohledů, je na ně upozorněno a jsou o nich zaznamenány informace (tzv. *„Event Record“*). Množství hlášených události se filtruje a je jim přiřazena kategorie důležitosti. Podle souvislosti s dalšími událostmi a službami je vybrána vhodná forma reakce, zhodnocení a uzavření. Existuje několik typů událostí podle důležitosti:

- **Informace**: událost, která neznamená žádnou poruchu a nevyžaduje žádnou reakci.
- **Varování**: upozornění na možnou blížící se poruchu získáním hraniční hodnoty na některé konfigurační jednotce nebo službě.

- **Výjimka**: událost, která značí, že některá služba nebo konfigurační jednotka nefunguje normálně, předává se pro zpracování procesu správa incidentů.

Vstupem procesu jsou výsledky z návrhu služby a přechodu služby, a výstupem jsou záznamy událostí.

5.4.2 Správa incidentů

Incident je neplánovaná událost, která způsobuje přerušení služby IT nebo její snížení kvality. Výpadek konfigurační jednotky bez dosavadního dopadu na službu IT je rovněž považován za incident. Kromě zaznamenání incidentů má proces Správa incidentů za cíl zajištění co nejrychlejšího návratu k normálnímu provoznímu stavu (který je popsaný v SLA) a obnovy postižené služby s minimálními negativními dopady na business společnosti. Pro záznam a manipulaci s incidenty je vhodné mít nástroj pro správu incidentů.

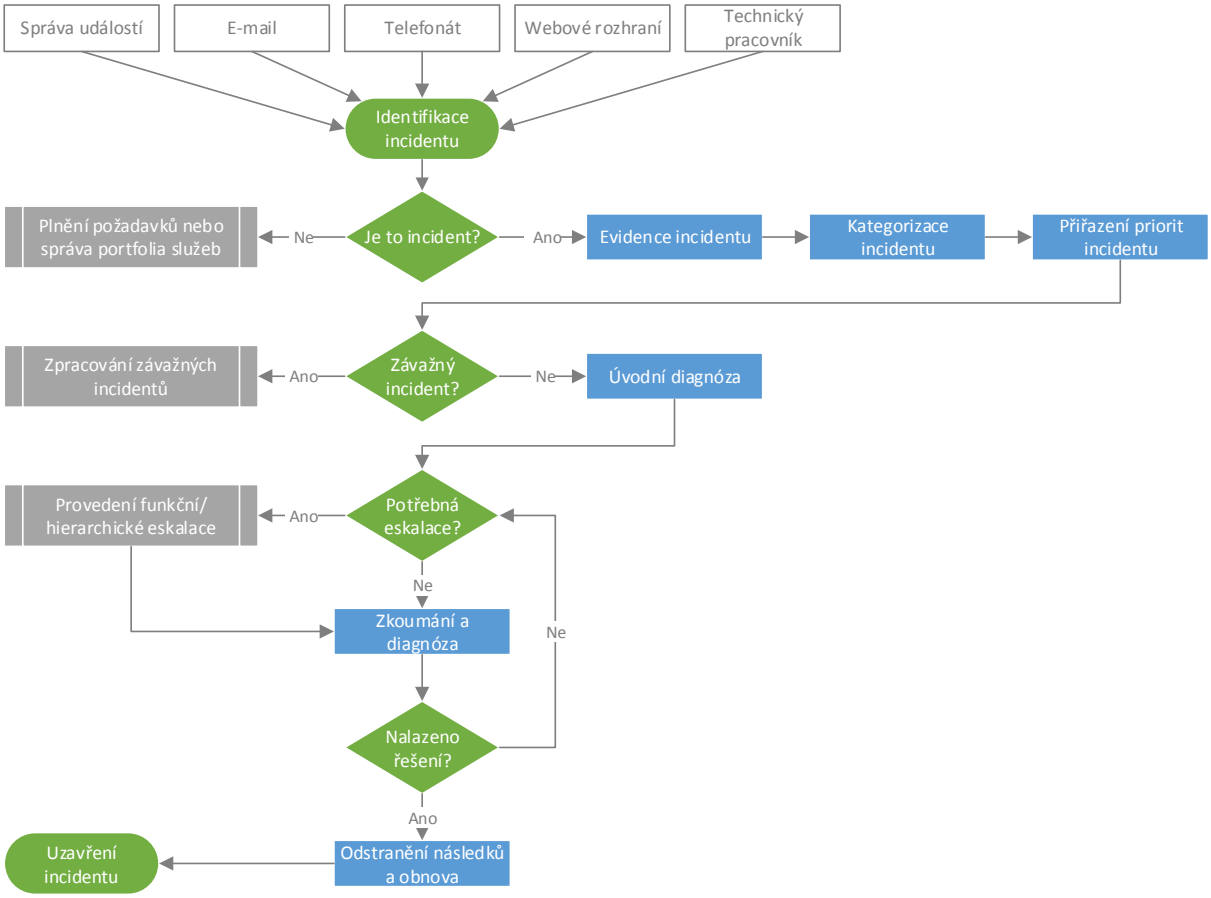
Incidenty jsou detekovány procesem Správou událostí nebo uživateli kontaktujícími organizaci a jsou zaznamenány do systému správy incidentů a vhodně kategorizovány. Každému incidentu je přiřazena priorita, která rozhoduje především o rychlosti dalšího zpracování. Úvodní diagnózu příčiny incidentu provádí pracovník Service desku (Kapitola 5.4.5) nejčastěji telefonicky hned při ohlášení poruchy pomocí modelu incidentů (opakující se poruchy jsou kategorizovány a obsahují popsané postupy pro jejich vyřešení). Pokud oprava neuspěje, je problém eskalován. Následuje další prozkoumání poruchy odborníky, identifikace poruchy, jeho řešení a obnova. Posloupnost jednotlivých aktivit zobrazuje *(Obrázek 8)*. Vstupem procesu jsou samotné incidenty a výstupem je obnovená služba a záznam o řešení incidentu.

5.4.3 Plnění požadavků

Požadavek na službu je zadáván od uživatele a představuje požadavek týkající se dodávaných služeb. Může to být dotaz na fungování aplikací, zjištění stavu řešení zadaného incidentu nebo rozšíření stávajících služeb. Jeho získání i celý životní cyklus je téměř shodný s procesem Správa incidentů. Zaznamenávány a spravovány jsou oddělením Service desk, které ručí za jejich vyřešení a korektní ukončení.

5.4.4 Správa problémů

Problém je příčinou jednoho nebo několika incidentů. Příčina může být známá, a tedy mluvíme o známé chybě, pro kterou již může existovat řešení (všechna existující řešení jsou popsány v databázi známých chyb), nebo neznámá. Proces Správa problému jednotlivé problémy nejprve identifikuje a odhaluje, následně je zaznamenává do záznamu problému spolu se všemi důležitými informacemi pro jejich vyřešení. Stejně jako incidenty, jsou problémy dále kategorizovány a jsou jim přiřazeny priority, dále jsou podstoupeny diagnóze pro porozumění příčině problému. Pokud se podařilo nalézt náhradní řešení, je důležité udělat záznam do databáze známých chyb. Řešení problému je následně implementováno, příčina problému vyřešena a celý problém je popsán a uzavřen. Kromě sledování celého životního cyklu problému se proces snaží učinit opatření, které pomáhají problémům předcházet. Vstupem procesu jsou záznamy problémů a incidentů a výstupem jsou vyřešené problémy, aktualizované záznamy o problému nebo RfC.



Obrázek 8: Posloupnost aktivit v rámci procesu Správa incidentů

5.4.5 Service desk

Oddělení Service desk v organizaci slouží jako kontaktní bod pro všechny uživatele IT služeb. Získává žádosti o službu od zákazníků a je místem pro hlášení poruch služeb. Zaznamenává a spravuje všechny incidenty, požadavky na službu a servisní požadavky, představuje první úroveň podpory IT organizace a slouží jako koordinační místo pro interní procesy a IT skupiny.

Service desk je obvykle podporován vhodným systémem, který slouží k zaznamenání i správě všech firemních požadavků a incidentů. Zaměstnanci Service desku mají za úkol požadavky sbírat a zadávat, ale taky správně kategorizovat a přiřazovat jim priority. V průběhu řešení požadavku informují uživatele o stavu řešení, po ukončení a přesvědčení se, že je uživatel s řešením spokojen, požadavky/incidenty uzavírají s vhodným popisem.

Organizační struktura Service desku je závislá na velikosti a struktuře organizace, její historii nebo jazycích, obecně lze však rozlišit několik struktur:

- **Lokální service desk**: pro každé pracoviště dané organizace je zřízen vlastní oddělení, tedy ve stejném místě spolu s uživateli.
- **Centrální service desk**: pro všechny pracoviště existuje pouze jediné oddělení.
- **Virtuální service desk**: části oddělení service desku jsou rozmístěny do různých míst, uživateli se však oddělení jeví jako jeden celek.

- **Nepřetržitý service desk**: speciální forma virtuálního service desku, která pro uživatele různých časových pásem zajišťuje nepřetržitou podporu.

5.4.6 Správa provozu IT

Proces Správa provozu IT zodpovídá za každodenní činnosti, důležité pro udržení správy služeb a pro podporu infrastruktury IT. Díky udržované stabilitě infrastruktury je pak organizace schopná zákazníkům dodat služby na dohodnutých úrovních. Dílčí činnosti Správy provozu služeb se zabývá řízením provozu IT a správou zařízení. První ze jmenovaných, provádí rutinní denní úkoly a zajišťují monitorování a kontrolu provozních aktivit. Správa činností zodpovídá za správu fyzické okolí výpočetních středisek, ve kterých se nachází infrastruktura IT (chlazení a napájení systémů, zadávání přístupových práv apod.)

5.5 Neustálé zlepšování služeb

Poslední fáze životního cyklu služby a taky pátá kniha ITIL, Neustálé zlepšování služeb (*„Continual Service Improvement“*, CSI), se zabývá udržováním dodané hodnoty zákazníkovi pomocí neustálého vyhodnocování a vylepšování kvality dodávaných služeb a průběžné přizpůsobování služeb IT měnícím se obchodních požadavkům. Pracuje na zlepšení každé z fází životního cyklu vývoje služby, procesů, technologií a všech souvisejících činností. Zefektivněním vývoje se snižují požadavky na zdroje během vývoje a snižují se náklady na poskytování služeb, které lze uplatnit pro zvýšení kvality dodávané služby. Ideální je snížit chybovost během procesu vývoje, zbavit se aktivit, které nepřispívají pro vývoj služby a zavést co nejvíce automatismů.

Model CSI je založen na nepřetržitém cyklu, který obsahuje šest kroků:

1. Plánování vizí založených na obchodních podmínkách.
2. Zhodnocení současné situace.
3. Pochopení priorit zlepšování podle dříve stanovených vizí.
4. Vytvoření plánu pro zajištění implementace zlepšení.
5. Zajištění za pomocí měření a metrik, jestli se dosáhlo plánovaného zlepšení.
6. Zajištění udržení probíhajícího zlepšování v rámci organizace.

Pro vykonávání jakýchkoliv změn a zlepšování je velmi důležité mít souhlas od vedení společnosti. Pro evidenci a dokumentaci různých zlepšení slouží Registr CSI.

5.5.1 Měření služeb

Aby bylo měření účinné a efektivní, je důležité nejprve stanovit počáteční hodnoty, které budeme později porovnávat s novými naměřenými hodnotami a z rozdílu určíme, zda má zkvalitňování služeb spíše klesající nebo stoupající tendenci. Důvody pro měření a monitorování služeb jsou potvrzení předcházejících rozhodnutí, nasměrování činností pro dosažení definovaných cílů, zdůvodnění konkrétního postupu a zasáhnutí ve vhodném bodu. Výsledky měření přispívají k výpočtu zlepšovacího procesu v sedmi krocích.

Základem pro měření služeb jsou metriky. Metrika je systém ukazatelů pro hodnocení efektivnosti a výkonnosti. Existují tři typy metrik, pomocí nichž se sbírají data pro CSI: technologické metriky, procesní metriky a metriky služeb. Technologické metriky jsou např. datová propustnost, dostupnost,

nebo výkonnost. Procesní metriky jsou shromažďovány ve formě kritických faktorů úspěchu a klíčových ukazatelů výkonnosti. Metriky služeb jsou výsledky z pohledu koncového uživatele, pro jejich výpočet jsou využívány metriky komponent.

Po naměření služeb a vyhodnocení výsledků měření jsou vytvořeny zprávy, které jsou předány businessu a slouží pro vyhodnocení úrovně dodávaných služeb. Zajímavé pro business jsou především ty zprávy, které dokumentují proběhlé události, které mohou mít špatný dopad na dodávání služeb. Politika organizace specifikuje rámec pro dodávání těchto zpráv pro jejich cílovou skupinu.

5.5.2 Metody a techniky CSI

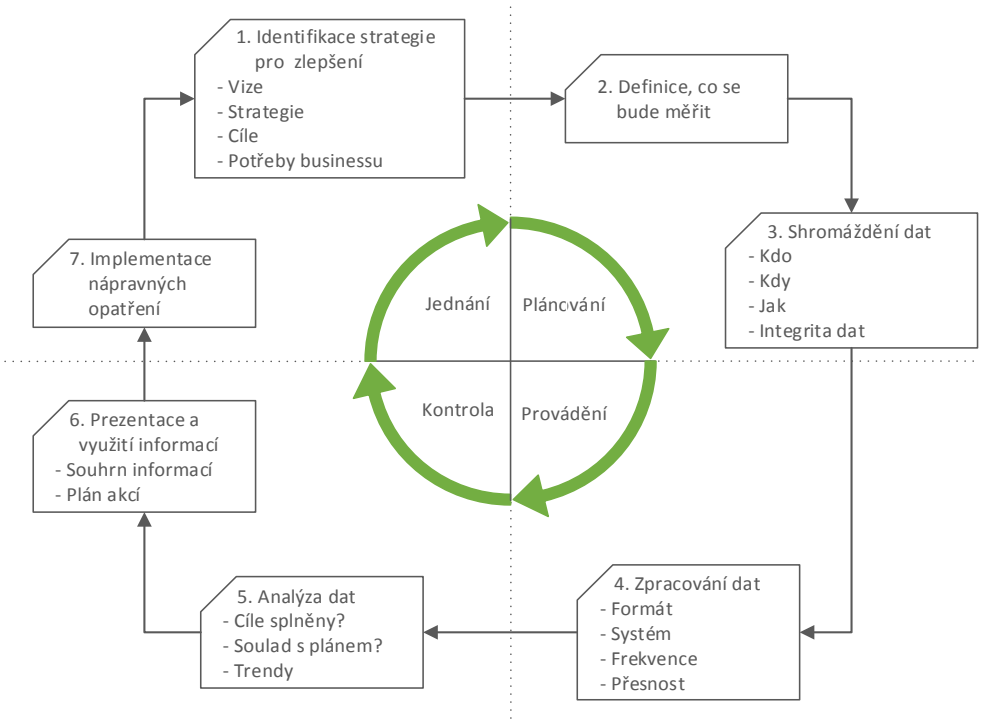
Pro neustálé zlepšování dodávání služeb zákazníkům proces používá mnoho metod a technik, patří k nim benchmarking, posuzování, Demingův cyklus a systém ukazatelů výkonnosti podniku.

Benchmarking je nepřetržitý systematický proces porovnávání a měření procesů, metod a produktů vlastní organizace s těmi, kdo byli uznáni jako vhodní pro toto měření, za účelem definovat cíle zlepšování vlastních aktivit [16]. Porovnávají se podstatné aspekty s vybranými benchmarky (stav objektu zaznamenaný v určitém čase), které by měly být nastaveny na vysokou výkonnost.

Posuzování je porovnávání procesního prostředí organizace s výkonnostními standardy. Slouží k otestování určitých postupů nebo procesů organizace pro jejich nedostatky.

5.5.3 Zlepšovací proces v sedmi krocích

Cílem procesu je příprava postupů pro identifikaci, definici, shromáždění, analýzu a plánování implementace zlepšení. Zahrnuje celkem sedm kroků, které zajistí sběr smysluplných dat pro vyhodnocení a provedení vylepšení. Jednotlivé kroky procesu znázorňuje *(Obrázek 9)*.



Obrázek 9: Popis jednotlivých kroků Zlepšovacího procesu v sedmi krocích

6 Modelování podnikových procesů

V této kapitole se budeme velmi stručně zabývat možností modelování podnikových procesů, popíšeme si existující nástroje a budeme diskutovat jejich vhodnost použití. To především ve spojitosti s tématem této práce, tedy modelování procesů v procesně založené společnosti používající ITIL a vhodnost modelů procesů pro společnost, která zavádí nebo používá agilní metodiku Scrum.

Podnikový proces je nástroj pro popis a pochopení toho, jak to ve firmě funguje. Má jasně daný vstup a výstup a je chápán jako souhrn činností a aktivit, které jsou vykonávány, aby byl splněn podnikový cíl. Tyto aktivity mají jasně dané pořadí, v jakém se mají vykonávat. V souvislosti s podnikovými procesy je důležité zmínit tzv. *„Business Process Management“*, což je komplexní přístup k řízení propojování interních procesů se strategií organizace, k analýze, optimalizaci a rozhodujících procesů. Procesně řízená organizace je zaměřena na výstupy ze svých klíčových procesů a na uspokojení potřeby zákazníka. Kromě reakcí na změny je i sama provádí, rozlišuje datově náročné procesy a procesy, které vyžadují volnost (např. komunikace s veřejností, sociální rozvoj atd.). Přínosy procesního řízení jsou především transparentnost všech procesů a postupů společnosti, flexibilita v provádění interních změn, zvýšená účinnost procesů, redukce nákladů procesů atd. [13]

Pro větší firmu, obsahující více firemních aktivit, je důležité zajistit, aby tyto aktivity byly prováděny efektivně a v co nejkratší době s minimálními náklady. Jelikož dnes většina firem zavádí procesní přístup na většinu prováděných operací, je nutné tyto procesy umět korektně popsat. Protože slovní popis nemusí být plně dostačující, je vhodnější procesy zobrazovat graficky. Model procesu je defakto šablona, která zobrazuje postupy shodné pro typově stejné procesy, popis konkrétní situace se nazývá instance modelu. Díky procesnímu modelu můžeme snadno posoudit stavy procesů, navrhnout nové nebo upravit existující a odstranit nepotřebné. Proces modelujeme jako soubor po sobě následujících činností, které vznikají na základě definovaných podnětů – vnitřních nebo vnějších. Každou z činností lze modelovat jako samostatný proces.

6.1 BPMN

Standard pro grafickou reprezentaci podnikových procesorů v diagramech popisuje BPMN (*„Business Process Modelling Notation“*). Byl vyvinut pro sjednocení způsobu popisu podnikových procesů, navržen tak, aby jeho popisu porozuměli různé skupiny lidí od analytiků a vývojářů až po vedení společnosti. Dalším cílem normy je podpora transformace BPMN do modelu popsaném v jiném jazyce pro výměnu modelů mezi různými procesními nástroji nebo interpretaci modelů pomocí workflow systému.

Norma definuje čtyři základní kategorie prvků, podle jejich použití v diagramu:

1. Elementy toku: události, aktivity, brány
2. Plavecké dráhy: bazén, dráha
3. Spojovací elementy
4. Artefakty: datové objekty, textové popisy, skupiny

Elementy toku jsou základními stavebními prvky procesu, pomocí nichž jsou popsány všechny důležité situace v procesu. Dělí se na události, aktivity a brány. Událost vyjadřuje stav procesu, pomocí kterého lze vyjádřit pořadí aktivit v procesu. Dělení událostí včetně jejich grafického popisu ukazuje

(Obrázek 10). Norma dovoluje upřesnit typ události zobrazením názorného znaku uvnitř kolečka (např. obálka pro přijetí zprávy, hodiny pro uplynutí určitého časového intervalu apod.).

Obrázek 10: Události

Pomocí aktivit modelujeme činnosti prováděné během procesu. Aktivity dělíme na procesy, sub-procesy a úkoly (dále nedělitelný proces). Sub-proces modeluje složenou aktivitu popsanou vnořeným procesem (v případě, že není nutné sub-proces dále popisovat, do symbolu aktivity se zobrazí „+" viz *(Obrázek 11)*).

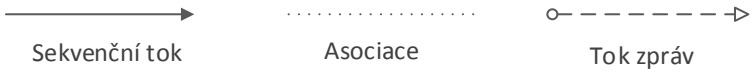
Obrázek 11: Aktivity

Brány v diagramech slouží ke spojování nebo rozdělování větví procesu. Pomocí ní vybíráme sekvenci, která se bude dále provádět. Existuje datový a událostmi řízený XOR, AND, OR nebo komplexní brána (význam jednotlivých zkratek odpovídá operátorům v Booleově logice), jejich symboly zobrazuje *(Obrázek 12)*.

Obrázek 12: Brány

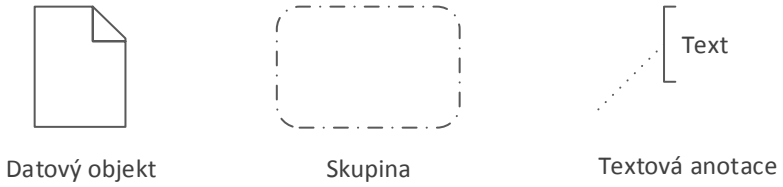
Plavecké dráhy umožňují zachycení a zobrazení modelovaných procesů z různých úhlů pohledů entit, podílejících se na procesu. Pro vymezení procesu se používá element bazén, který může být dále dělen na více drah. Dráhy vyjadřují zodpovědnosti v nich obsažených aktérů a jednotek. Bazén i dráhy jsou vyjádřeny obdélníkem s ostrými hranami.

Spojovací elementy propojují elementy toku a artefakty. Norma definuje tři druhy: sekvenční tok (jednoznačné pořadí), asociace (připojení diagramu k popisovanému prvku) a tok zpráv (interakce mezi různými procesy), jejich značení zobrazuje *(Obrázek 13)*.



Obrázek 13: Spojovací elementy

Artefakty umožňují specifikovat doplňkové informace procesu, jejich použití však nijak neovlivňuje provádění procesu. Mezi artefakty patří: datové objekty (používané dokumenty), skupiny (označují množinu elementů pro dokumentační význam) a textové anotace *(Obrázek 14)*.



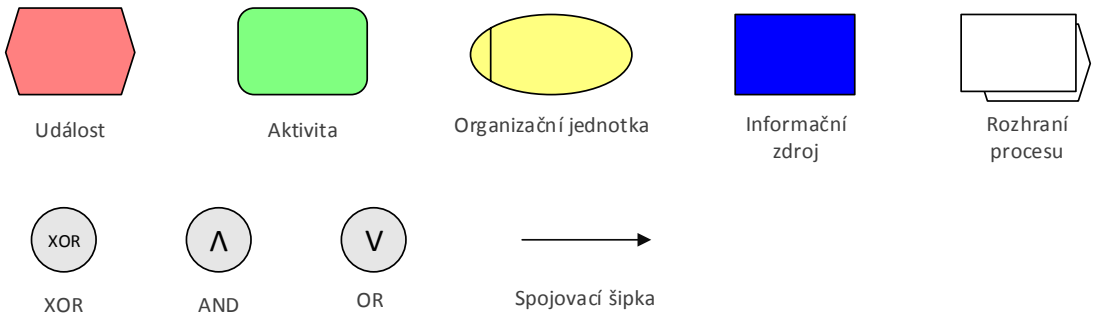
Obrázek 14: Artefakty

Hloubka úrovně popisu jednotlivých procesů je dána složitostí a požadovanou detailností modelovaných procesů [17].

6.2 Metodika EPC

Model EPC („*Event-driven process chain*“, diagram procesu řízeného událostmi), který zobrazuje zřetězené události a aktivity do posloupnosti realizující požadovaný cíl.

Model se skládá z událostí, aktivit, spojovacích šipek, logických spojek (AND, OR, XOR) a rozhraním procesu (definuje odkaz na jiný proces). Význam uvedených elementů je stejný jako u normy BPMN, liší se pouze jejich symboly *(Obrázek 15)*. Dále může být model rozšířen o element organizačních jednotek, který rozšiřuje element aktivity; a o element informačních zdrojů, který obsahuje využívané informace v rámci aktivit [13].



Obrázek 15: Symboly EPC metodiky

6.3 Nástroje pro modelování procesů

Softwarových nástrojů pro modelování procesů existuje celá řada. Zde uvedeme ty, které se obecně nejvíce používají s ohledem na možnost dalšího použití konkrétních nástrojů ve vybrané společnosti – Microsoft Visio a ARIS. Další z existujících nástrojů pro tvorbu modelů procesů: System Architect (IBM), PowerDesigner (Sybase).

6.3.1 Microsoft Visio

Komerční nástroj společnosti Microsoft, Microsoft Office Visio (dále jen MS Visio) určeného pro vývoj na platformě Windows. Primárně tento nástroj není určen pouze pro modelování podnikových procesů, přesto pro tuto oblast nabízí poměrně solidní možnosti modelování. Dle mého názoru, pro uživatele, který někdy pracoval s kterýmkoliv jiným nástrojem z řady Microsoft Office, není nutné delší školení, protože nástroje v MS Office jsou poměrně intuitivní. Pro modelování lze vybrat z celé řady

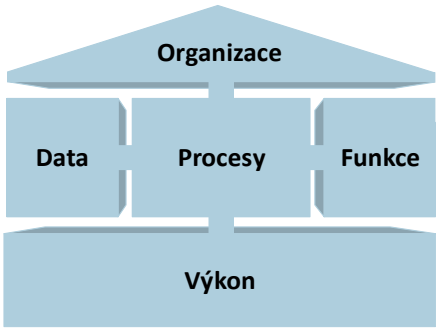
tvarů od elektrických sítí, návrhu organizační struktury, rozvržení nábytku v kancelářích, časových diagramů, návrhu databázových modelů až k vývojovým diagramům (pozn. všechny obrázky a grafy, které jsou součástí této práce, byly vytvořeny právě v nástroji MS Visio). Nástroj bezproblémově sdílí data se všemi aplikacemi s řady Microsoft Office, výsledky lze exportovat do formátů: jpg., gif., png., bmp.,tiff., pdf., emf., svg., dwg., dxf., atd.

Pro modelování procesů použitých v ITIL i pro zobrazení Scrum procesů jsou podstatné především diagramy z kategorie „Vývojový diagram“ (Flow chart) a „Obchod“ (Business). Obsahuje i předpřipravené symboly pro tvorbu modelů v BPMN a EPC. Další z možných použitých modelů je základní vývojový diagram, diagram organizační struktury, síťový diagram (např. pro zobrazení IT infrastruktury) nebo diagramy modelů UML (diagram tříd, diagram případu užití, sekvenční diagram, diagram aktivity). Jednotlivé části procesu lze doplnit o bližší charakteristiky, které kompletně dokumentují celý proces. Příklad EPC diagramu spolu s hlavním pracovním oknem nástroje zobrazuje Příloha A *(Obrázek 30).*

6.3.2 ARIS

Komerční nástroj německé společnosti IDS Sheer přímo určen k modelování podnikových procesů. ARIS nástrojů existuje několik. ARIS Express, open source verze určená pro studenty vysokých škol, možnost stažení zdarma po zaregistrování na webové stránce www.ariscommunity.com/aris-express, její funkčnost je ořezána, pro základní modelování procesů však stačí. Placené verze: ARIS Business Architect, ARIS Business Publisher, ARIS Process Governance a další. V rámci kurzu „Strategické řízení informačních systémů“ na FIT VUT jsem měla možnost si vyzkoušet práci v nástroji ARIS Business Architect (dále pouze ARIS), proto se v dalším textu budeme bavit o práci právě v tomto nástroji. Na rozdíl od MS Visio, není práce v ARIS úplně intuitivní hned na první pohled a je nutné kratší proškolení o možnostech, které nástroj nabízí. ARIS podporuje multi-uživatelskou práci, možnost verzování, centrální úložiště, automatickou generaci do programového kódu.

S nástroje ARIS je přímo spojena metodika modelování procesů od prof. Augusta Scheera. Pro lepší pochopení, jaké všechny diagramy lze nástrojem modelovat a jak jsou v nástroji logicky řazeny, si metodiku velmi krátce popíšeme.



Obrázek 16: Pohledy metodiky ARIS

(Obrázek 16) popisuje pět základních pohledů, na kterých je založena metodika ARIS. Organizační pohled popisuje pracovníky a vztahy mezi nimi a organizačními jednotkami, datový pohled je popsán událostmi a stavy, které jsou zastoupeny daty. Procesní pohled zaznamenává vztahy mezi jednotlivými pohledy, funkční pohled je tvořen funkcemi systému a vztahy mezi nimi a výkonový pohled obsahuje prvky měření procesů a jejich metriky. Všechny pohledy jsou mezi sebou vzájemně propojeny a jejich kombinace pokrývají téměř všechny vlastnosti fungování organizace.

Možnosti modelování procesů přímo korespondují s metodikou ARIS, typy modelů jsou rozděleny do pohledů. Nástroj podporuje tvorbu modelů podle notace BPMN, EPC a celou řadu dalších typů modelů pro podnikové modelování procesů. Typ a rozsah detailů, možnosti formátování popisků a další nastavuje administrátor nástroje, což podporuje jednotnost popisu procesů. Příklad organizačního diagramu spolu s hlavním pracovním oknem nástroje zobrazuje Příloha A *(Obrázek* 31*)*.

6.4 Zhodnocení

Jelikož cílem této práce není hodnocení a porovnávání celé řady nástrojů pro modelování procesů, rozhodla jsem se popsat pouze dva hlavní, se kterými se ve svém okolí, i v rámci vybrané společnosti, setkávám. Nicméně bych ráda použila výsledky zhodnocení zmíněných dvou nástrojů z bakalářské práce Fakulty informatiky a statistiky Vysoké školy ekonomické v Praze z roku 2010 od Miroslava Fary [4]. Ten pomocí dotazníku zjišťoval od studentů, který z nástrojů jim pro tvorbu školních projektů, zaměřených právě na modelování podnikových procesů, vyhovuje lépe. Mezi kritérii hodnocení nástrojů byly mimo jiné i pořizovací cena produktu, nápověda, jazyková podpora, instalace a prostředí nástroje, podporované notace a metodiky, generování programového kódu a množství modelů. Přestože bylo hodnocení jistě velmi subjektivní, v testu lépe dopadl nástroj Microsoft Visio, nicméně oba nástroje byly vyhodnoceny jako velmi vhodné pro používání při modelaci podnikových procesů.

Pro tuto práci, tedy modelování procesů v rámci knihovny ITIL a metodiky Scrum, byl jako nástroj pro modelování procesů vybrán Microsoft Visio s přihlédnutím na mé zkušenosti spojené s tímto nástrojem, ale především na možnost dalšího využití namodelovaných procesů ve vybrané společnosti i pro širokou veřejnost (dle mého názoru, je Microsoft Visio rozšířenější nástroj a veřejně lépe dostupný než ARIS).

7 Integrace ITIL a Scrum v AVG

V předchozích kapitolách jsme se věnovali popisu procesní knihovny ITIL a agilní metodiky Scrum z teoretického hlediska, v této kapitole práce se zaměříme na vybranou společnost a možnosti integrace procesní knihovny ITIL a agilní metodiky Scrum v ní.

Společnost AVG Technologies si nepřeje zveřejňovat interní data a informace, proto tato kapitola není viditelná ve veřejné verzi práce.

8 Návrh metodiky SCRIL

Na základě informací získaných z předešlých kapitol − teoretického rozboru agilní metodiky Scrum, procesní knihovny ITIL a nástrojů pro modelování podnikových procesů, bude v této kapitole navržena metodika, která spojí přístupy Scrum a knihovny ITIL do jedné fungující metodiky. V předešlé kapitole jsme analyzovali situaci ve společnosti AVG Technologies, která pro vývoj používá kombinaci obou přístupů. Byly vyhodnoceny pozitivní dopady sjednocení obou přístupů, ale bylo poukázáno i na negativní dopady a s tím spojené problémy. Především z těchto poznatků jsem při navrhování nové metodiky vycházela. Metodiku jsem nazvala **SCRIL**, jako výsledek spojení slov Scrum a ITIL.

Metodika SCRIL byla navržena tak, aby splňovala požadavky na metodiky cílové skupiny firem poskytující IT služby, kterékoliv velikosti. Je využitelná pro vývoj nových i rozšiřování existujících systémů vyvíjených pomocí nějaké metodiky nebo bez ní. Při návrhu byl kladen největší důraz na její dostatečnou obecnost, díky které bude její implementace možná v celé řadě firem.

8.1 Role

V rámci metodiky SCRIL existuje řada rolí, které se účastní vývoje software a s ním spojených aktivit. Jedna osoba může zastávat více rolí, stejně tak jednu roli může zastávat více osob. Jejich přidělení závisí na profesních zkušenostech a vlastnostech dané osoby. Definujeme jen role nutné pro potřeby metodiky, předpokládá se, že budou doplněny o další množství rolí pro potřeby konkrétní firmy. U každé role bude přesně uvedeno, jestli vychází pouze z metodiky Scrum nebo knihovny ITIL nebo je spojením obou metodik.

8.1.1 Scrum master

Role Scrum mastera je velmi důležitá, protože právě on řídí vývoj scrumového týmu. Pojem „řídí“ ovšem v tomto případě neznamená direktivní způsob řízení týmu lidí, spíše jako jejich podpora a koučování. Scrum master dává pozor, aby všechna daná pravidla byla dodržována, případně pomáhá týmu je dodržovat. Probírá s nimi stávající a nové metody v práci, dodává jim motivaci a podporuje jejich tvůrčí činnost. Pokud nastane jakýkoliv problém v týmu, je na něm, aby nalezl nejlepší možné řešení a pomohl ho realizovat. Stará se o to, aby jednotliví členové týmu nebyli vyrušování od práce, působí jako spojka mezi týmem a zbytkem firmy. Zároveň může zastávat i roli inženýra/vývojáře a se členy svého týmu sdílí jednu místnost.

Řídí schůzky, které jsou součástí metodiky. Zajišťuje, aby byl dodržovaný jejich časový harmonogram, aby během jejich konání byly probírány pouze věci k tématu a účastnili se jen požadovaní lidé. Role Scrum mastera vychází čistě z metodiky Scrum. V případě implementace pouze knihovny ITIL by se dalo říct, že podobnou roli v týmu zastává Team leader, který vede a řídí svůj tým. V případě, že je potřeba role Scrum mastera i Team leadera, je rozdělení rolí důležité mezi dvě osoby s vymezenými povinnostmi.

8.1.2 Team leader

Role Team leader má na starost řízení určitého týmu. Zajišťuje jeho personální složení, ohodnocení práce členů v týmu, jejich motivování a odpovídá za plnění cílů. Diskutuje s vedením činnosti v týmu a další důležité skutečnosti, předává tyto informace členům týmu. Pro tuto roli jsou podstatné dobré komunikační schopnosti, umění efektivního využívání času a dobrých organizačních vlastností.

Důležitá skutečnost u této role je ta, že člověk, který ji vykonává, by zároveň neměl zastávat i roli Scrum mastera ve stejném týmu, neboť jejich působení v týmu má mít jiný význam pro jeho členy. Zatímco Scrum master lidi podporuje a pomáhá jim zvládnutí lepších technik vývoje a zajímá jej činnost pro přidání hodnoty produktu, Team leadera zajímá spíše kvalita práce členů, na základě které udržuje personální složení týmu. Není podmínkou, aby Team leader sdílel stejnou kancelář jako členové jeho týmu. Jak již bylo řečeno, role Team leadera, tak jak je zde popsaná, je definovaná pouze v knihovně ITIL.

8.1.3 Inženýr/ Vývojář

Role inženýra neboli vývojáře označuje člověka, který v týmu přímo vyvíjí daný produkt. Každý vývojový tým ideálně obsahuje 6−10 inženýrů. Každý inženýr zastává roli programátora i testera zároveň. Role programátora nese zodpovědnost za softwarovou architekturu produktu spolu s jeho konkrétní implementací. Předpokládá se znalost používané platformy a návrhu algoritmů. Rovněž je zodpovědný za zpracování programové části dokumentace. Role testera nese zodpovědnost za provádění testů funkčních částí produktu. Předpokládá se znalost testovacích postupů a jejich efektivní provádění. Jejich spojení do jedné osoby zamezuje neefektivnosti práce v týmu. Pokud by to tak nebylo, testeři by na začátku museli čekat, než programátoři vyvinou část produktu, a teprve až pak by ji mohli začít testovat – tedy by jejich kapacita nebyla plně využitá. V případě, že inženýr zastává obě role, od začátku všichni členové vyvíjí části produktu a poté je všichni postupně testují. Aby testování bylo úplné a správné, neměl by žádný inženýr testovat svoji vyvinutou část. Tento přístup nicméně výrazně zvyšuje profesní požadavky na roli inženýra. V případě, že firma zaměstnává lidi profesně zaměřené buďto na testování nebo programování, je nutné jejich proškolení nebo náhrada za odborně zdatnější zaměstnance. Jak již bylo řečeno, všichni inženýři jednoho týmu sdílejí spolu se Scrum masterem jednu kancelář a dle jeho pokynů dodržují dané pravidla.
Role vývojáře, tak jak je zde popsaná, vychází čistě z metodiky Scrum a to kvůli spojení role programátora a testera do jednoho člověka. Knihovna ITIL obě role rozděluje mezi dva lidi a seskupuje je do týmů jen testerů a jen programátorů, které mají svého Team leadera.

8.1.4 Vlastník produktu

Role Vlastníka produktu nese zodpovědnost za všechny součásti, které produkt obsahuje. Vytváří rámcový plán vývoje produktu, určuje nástroje analýzy. Předpokládá se, že každá firma má svůj vývoj rozdělen na více produktů, kdy každý z nich obsahuje činnosti napříč více týmy a platformami, které je nutné zajistit. Produkt může mít více vlastníků a vlastník může řídit více produktů. Osoba, která tuto roli zastává, získává informace od oddělení businessu, které je v přímém spojení se zákazníky a uživateli. Všechny tyto požadavky shromáždí do Produktového backlogu, přiřadí jim kategorii a prioritu.
Kromě businessu spolupracuje s těmi vývojovými týmy, které jsou přiřazeny pro daný produkt. Na pravidelných schůzkách s nimi diskutuje obsah Produktového backlogu a zajišťuje, aby všichni členové týmu rozuměli všem požadavkům. Diskutuje s týmem a Scrum masterem, které požadavky z Produktového backlogu budou přesunuty do Backlogu sprintu. V podstatě je součástí vývojového týmu a ideálně se zbytkem jeho členů sdílí jednu kancelář. Pokud na vývoji jednoho produktu pracuje více vývojových týmů, udržuje se všemi velmi blízký kontakt, aby jim mohl předávat zpětnou vazbu od zákazníků a motivovat je tak k další práci.
Roli Vlastníka produktu popisují obě metodiky, jak Scrum, tak ITIL. Výše uvedený popis činností odpovídá spíše scrumovému popisu vlastníka, hlavně díky aktivitám spojených s backlogy a spojení s týmy. Knihovna ITIL roli Vlastníka produktu jako takovou nepopisuje, přesto je v ní zmíněna role

Business analytika, která spadá do oddělení businessu, které se zabývá komunikací se zákazníky a předává jejich požadavky oddělením programátorů – podstata role je tedy stejná.

8.1.5 Uživatel

Role uživatele představuje roli koncového uživatele, který konzultuje produkt s vyvíjející firmou během jeho vývoje. Předpokládá se, že uživatel je dostatečně proškolený k tomu, aby mohl produkt efektivně využívat. Role uživatele je stejná jak pro metodiku Scrum, tak ITIL.

8.1.6 Zadavatel

Role zadavatele představuje zákazníka, který požaduje (objednává) řešení a je zodpovědný za schvalování vytvářených částí produktu. Může se jednat o externího zákazníka objednávajícího „řešení na zakázku“ nebo zákazníka – uživatele, který požaduje změnu dosavadního dodaného řešení. Kromě externího zákazníka lze uvažovat i zákazníka interního (tedy zaměstnance firmy). Ten může jako požadavek zadat např. zjištěnou závadu ve firemním systému nebo vyvíjeném produktu, nebo požadavek na vylepšení stávající části systému/produktu, která vyžaduje implementaci. Role zadavatele je stejná jak pro metodiky Scrum, tak ITIL.

8.2 Artefakty

Počet artefaktů pro metodiku SCRIL je omezeno na co nejmenší možné množství. Každá firma jejich počet reguluje dle vlastních možností, obecně ale platí, že jejich tvorba a údržba nesmí kriticky zdržovat členy týmu od jejich skutečné práce. Udržují se jen ty artefakty, které jsou opravdu důležité pro dodání produktu a splnění požadavků businessu. V této části si popíšeme ty artefakty, jejichž tvorba a údržba jsou klíčové pro metodiku SCRIL. U každého z nich bude rozlišeno, jestli je převzat z metodiky Scrum nebo ITIL.

8.2.1 Produktový backlog

Získané požadavky, jejichž vývoj je složitější a je nutný naplánovat, jsou roztřízeny podle kategorií a přiřazeny určitému produktu v Produktovém backlogu. Vlastník produktu ohodnotí požadavky a vznikne tak seznam seřazený podle priorit. Spolu se zadavatelem ke každé položce definuje kompletní popis s časovým odhadem na vývoj. Stanovené priority nejsou závazné, naopak je žádoucí, aby se s časem měnily, protože to ukazuje, že nově získané informace byly zakomponovány do rozhodovacího vývoje výběru. Hlavní zodpovědnost za Produktový backlog nese Vlastník produktu, který zajišťuje, aby mu všichni členové týmu porozuměli a byl pro ně kdykoliv viditelný. Pro všechny týmy, které se zabývají vývojem konkrétního produktu, existuje jediný Produktový backlog, jehož položky si rozdělují mezi sebe s ohledem na zdroje a čas.
Tento artefakt je převzat z metodiky Scrum. ITIL jeho existenci nezmiňuje, přesto se předpokládá, že programátoři mají seznam položek, které je nutné vyvinout ve firemním systému (žádné speciální pravidla se na něj ale nevztahují).

8.2.2 Backlog sprintu

Z Produktového backlogu se na plánovací schůzce po domluvě se Scrum masterem a vývojovým týmem vyberou položky s největší prioritou a přesunou se do Backlogu sprintu. Obsah Backlogu sprintu

je tak v každém sprintu jiný a představuje aktuální seznam položek, které vývojový tým vyvíjí. Hlavní odpovědnost za Backlog sprintu nese Scrum master, který vede členy svého týmu k úspěšnému dokončení naplánovaných věcí. Po domluvě s týmem a s Vlastníkem produktu lze z Backlogu sprintu položku odstranit, resp. vrátit zpět do Produktového backlogu. Backlog sprintu má každý tým právě jeden, nedělí se o něj s žádným dalším týmem.
Podobně jako u Produktového backlogu, i Backlog sprintu je převzat čistě z metodiky Scrum.

8.2.3 Operativní backlog

Získané požadavky, jejichž vývoj není nikterak složitý a zadání vyžaduje poměrně rychlé řešení, jsou přiřazeny do Operativního backlogu. Jednotlivé položky jsou v Operativním backlogu uspořádány podle priorit, podobně jako v Produktovém backlogu. Opět se předpokládá, že se priority s časem budou měnit a seznam se bude aktualizovat. Existuje jeden hlavní Operativní backlog, do kterého směřují všechny požadavky splňující podmínky pro operativní položky. Požadavky jsou v něm rozděleny podle skupin, kterým pro vývoj připadají. Scrum masteři daných skupiny přebírají požadavky a předávají je do Operativních backlogů scrumových týmů (každý scrumový tým má vlastní Operativní backlog). Položky z nich jsou postupně zpracovávány podle priorit.
Tento artefakt přímo nedefinuje ani metodika Scrum, ani ITIL, nicméně nutnost jeho použití ve větších společnostech časem ukáže sama praxe. Můžeme ho považovat za obdobu Produktového backlogu s méně k němu se vztahujícími činnostmi.

8.2.4 Operativní release backlog

Naimplementované a otestované položky z Operativních backlogů scrumových týmu jsou vývojáři přesunuty do Operativního release backlogu. Jednotlivé položky jsou v něm uspořádány primárně podle požadovaného nejpozdějšího data vydání a sekundárně podle priorit. Kromě dat přidává vývojář i informaci o tom, které technické kroky je nutné udělat pro to, aby byla implementovaná položka úspěšně vydaná do produkce. Vlastníkem backlogu je Release manager, který zodpovídá za všechny činnosti, které s ním souvisí.
I tento artefakt není převzatý z žádné z obou metodik. Jeho použití se zdá být přínosné po zkušenostech z praxe, přináší přehled a pravidla pro zpracování požadavků před zavedením do release. Jeho účinnost bude otestována a zhodnocena.

8.2.5 Produktový release backlog

Do Produktového release backlogu jsou směřovány všechny naimplementované a na vývojovém prostředí otestované položky z Produktových backlogů. Položky v něm jsou, stejně jako u Operativního release backlogu, uspořádány primárně podle požadovaného data vydání, pak podle priorit. Protože se jedná o položky, jejichž implementace byla komplexnější než u operativních, musí být ke každé z nich přidán velmi detailní soupis všech kroků nutných pro korektní a úplné nasazení do produkce spolu se všemi důležitými informacemi. Kromě toho musí být dokládán i záznam o tom, že testy na vývojovém, popř. testovací prostředí proběhly v pořádku a položka skutečně může být vydána do produkce. Vlastníkem backlogu je Release manager.

Obdobně jako u Operativního release backlogu, i tento artefakt je na základě zkušeností z praxe navrhnut jako nový, tedy ani Scrum, ani ITIL jej nepopisují. Jeho účinnost bude otestována v praxi.

8.2.6 Scrum board

Obsah Backlogu sprintu vhodně zobrazuje tzv. Scrum board (v překladu Scrum tabule). Ta je rozdělena na tři hlavní sloupce: ToDo, In Progress, Done; a řádky podle počtu vývojářů. Sloupec „ToDo“ obsahuje položky z Backlogu sprintu přiřazeny konkrétnímu vývojáři, které se musí udělat. Sloupec „In Progress“ obsahuje položky, které konkrétní vývojář právě řeší, a sloupec „Done“ obsahuje ty položky, které jsou naimplementované i otestované. Tabule může být řešena buďto hmotnou tabulí, na kterou si vývojáři připichují lístečky se svými položkami a snadno tak vidí, kolik práce jim zbývá a kolik už mají hotového; nebo lze tabuli používat virtuálně v tomu přizpůsobeném systému. Výhodou a nutností dodržování pravidla používání tabule je to, že kdykoliv se kdokoliv na ni podívá, zjistí, jestli tým je schopen naplánovanou práci stihnout či ne a na čem zrovna pracuje. Malování lístečků a připínání do sloupců zvyšuje kreativitu vývojářů a podporuje to jejich další práci na projektu. Mimo obsah Backlogu sprintu lze provozovat i tabuli, která bude zobrazovat obsah Produktového backlogu. Ovšem v týmech, kde je Produktový backlog velmi obsáhlý lze udržování aktuálnosti tabule považovat za neefektivní a časové velmi náročné.
Tento artefakt je převzat čistě z metodiky Scrum, která jej považuje za jednu ze svých hlavních vlastností, díky které je hodnota Scrum přinášena týmu a společnosti. Knihovna ITIL existenci Scrum boardu nezmiňuje, dá se ale předpokládat, že podobnou tabuli by programátoři používali i při vývoji požadavků ve společnosti používající jenom ITIL, protože umožňuje jednoduchý přehled, co ještě zbývá udělat.

8.2.7 Burn Down graf

Stejně jako Scrum Board, je i Burn Down graf možnost, jak vizualizovat průběh a výsledky sprintu. Graf na ose x znázorňuje čas sprintu a na ose y časové ohodnocení všech požadavků pro aktuální sprint. V grafu by měly být aktuální časové ohodnocení všech nehotových úkolů, takže by měl být schopný ukázat předpokládaný konec sprintu – jestli se dokončení sprintu zpožďuje nebo předbíhá oproti naplánovanému průběhu. Ideálně by měl být graf automaticky generovatelný systémem, ve kterém tým spravuje Produktový backlog. V případě, že to tak není, je na zvážení Scrum mastera velikost backlogu a práce strávená tvorbou a aktualizováním grafu.
Artefakt ve smyslu, jak je zde popsán, vychází z metodiky Scrum, jelikož slouží k zobrazení výsledků sprintu. Knihovna ITIL jej přímo nezmiňuje, ale opět se předpokládá, že podobný graf, znázorňující průběh a výsledky vývoje programátorů ve vývojových týmech je rovněž používán.

8.3 Zpracování požadavků

Protože se metodika SCRIL zabývá propojením metodiky vývoje Scrum a procesní knihovny ITIL, je nutné životní cyklus produktu pojmout z širšího hlediska než jen jako vývoj software. Životní cyklus začíná ve chvíli získání požadavku od zákazníka nebo zadáním nového požadavku ze strany vedení/businessu. Obecně se budeme věnovat pouze těm typům požadavků, jejichž zpracování probíhá skrze metodiku Scrum. Ve společnosti, která implementovala ITIL, lze jako požadavky, které je nutné implementovat považovat: incidenty, problémy, požadavek na změnu, požadavek na službu nebo požadavek na nový projekt/produkt. Každý z uvedených typů požadavků je zpracován v rámci samostatného procesu, jehož průběh a činnosti se více či méně liší od ostatních. Jednotlivé procesy budou postupně popsány se zaměřením na aktivity, které jsou spojeny s činnostmi metodiky Scrum.

8.3.1 Service desk

Klíčovým procesem/oddělením je Service desk, který zastává přímý kontakt se zákazníky a získává od nich požadavky a informace. Vlastník procesu je Team leader, který řídí všechny činnosti a spravuje organizační strukturu. Ta je rozdělena na dvě hlavní části. První část neboli první úroveň, představují pracovníci, jež poskytují přímou podporu pro zákazníky – telefonickou, přes mail, odpovídající formulář nebo na webových stránkách. Druhou úroveň jsou pracovníci, kteří mají na starost administrativu systémů firmy a jsou oprávněni měnit nastavení systémů, účtů apod. Důležitou součástí oddělení Service desku je mít k dispozici interní systém, který umožní zadání a sledování požadavků. Především musí pokrývat celý jejich životní cyklus a být použitelný pro celou společnost.

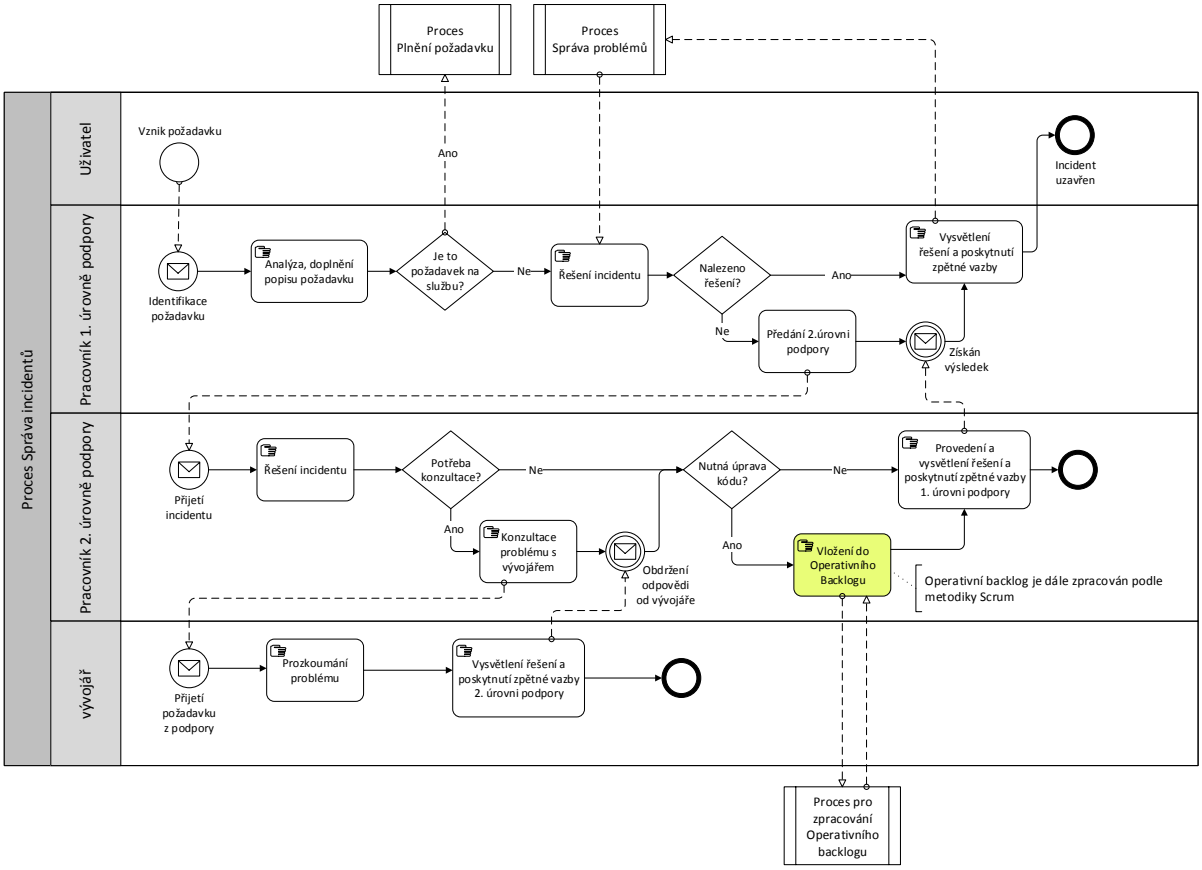
Oddělení Service desku pokrývá zpracování procesů z knihovny ITIL: Správu incidentů a Plnění požadavků. Po získání zadání od uživatele se hned v první fázi rozliší, zda jde o incident či požadavek na službu a podle toho se dále postupuje buď podle procesu Správa incidentů, nebo Plnění požadavků. Celé oddělení Service desku je zpracováno tak, jak jej popisuje knihovna ITIL. V této fázi vývoje není důležité, jakou metodikou budou požadavky zpracovány, proto proces není nijak ovlivněn pravidly a požadavky Scrum. Jediné omezení je to, že požadavky musí být spravovány vhodným systémem tak, aby bylo možné je dále použít pro účely metodiky Scrum (převedení do produktového backlogu, zobrazení na Scrum boardu apod.).

8.3.2 Správa incidentů

Vlastník procesu je Team leader Správy incidentů neboli Incident manager, často stejná osoba zastávající roli Team leadera oddělení Service desku. Ten má na starosti veškeré aktivity ohledně získávání a řešení incidentů. Mimo to stanovuje metriky pro vyhodnocení úrovně procesu a dělá pravidelné reporty, ve kterých udržuje informace o různých aspektech řešení incidentů. Ty jsou velmi důležité pro zhodnocení výkonnosti a efektivnosti celého procesu.

Vstupem procesu je incident nahlášený od uživatele přes oddělení Service desk nebo incident detekovaný přes proces Správa událostí. Získaný incident je zadán do firemního systému pro správu incidentů, kde je vložen jeho detailní popis a je vhodně kategorizován a je mu přidána priorita. Pokud se jedná o opravdu závažný incident, je provedená eskalace na odpovídající oddělení, které provede neodkladné kroky k okamžitému vyřešení incidentu. Pokud incident není závažný, provede k tomu určený zaměstnanec úvodní diagnózu, v rámci které zjistí, zda pro konkrétní incident neexistuje v systému pro záznam incidentů postup jeho řešení a jestli ano, provede dané kroky (spojitost s procesem Správa problémů). Jestli ne, provede potřebnou hierarchickou nebo funkční eskalaci. Po eskalaci je incident znovu prozkoumán odborníky a je navrhováno jeho řešení. V případě nutného programového řešení se tak zadání incidentu stává vstupem pro agilní metodiku Scrum, jejíž průběh si popíšeme v dalších kapitolách, vydání incidentu do produkce zajišťuje proces Správa releasů. Po vyřešení incidentu je jeho řešení popsáno ve vhodném rozsahu v záznamu incidentu a ten je poté uzavřen. Detailní popis jednotlivých kroků spolu s rolemi zobrazuje *(Obrázek 17)*.

Proces Správa incidentů je převzat z knihovny ITIL, popis jeho aktivity odpovídají z větší části popisu v ITIL – zpracování požadavku v první a druhé úrovni oddělení Service desku. Modifikace nastává ve třetí úrovni, ta je přímým vstupem do metodiky Scrum. Předání probíhá skrze vložení záznamu požadavku do Operativního backlogu, který je již zpracován kompletně podle metodiky Scrum. Jednodušší přechod zajišťuje vhodná volba firemního systému pro správu požadavků.



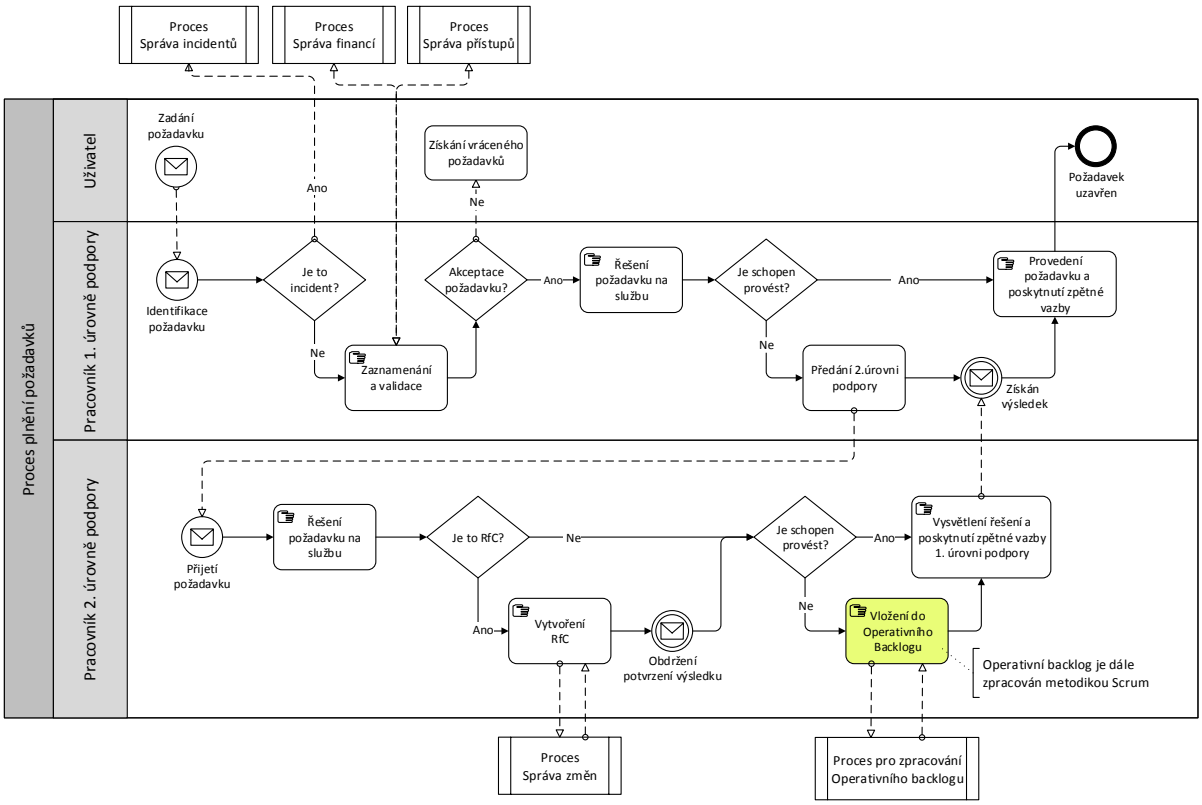
Obrázek 17: Diagram procesu Správy incidentů ve spojitosti s ostatními procesy (podle BPMN), zeleně vyznačená část označuje místo, kde proces z knihovny ITIL navazuje na metodiku Scrum

8.3.3 Plnění požadavků

Vlastník procesu je Team leader procesu Plnění požadavků, který většinou zastává i roli Team leadera oddělení Service desku. Má na starosti všechny aktivity týkající se získávání a vyřešení požadavků na služby. Provádí pravidelné reporty výsledků plnění požadavků a stanovuje metriky, na základě kterých vyhodnocuje výkonnost a efektivitu procesu.

Vstupem procesu je požadavek na službu nahlášený od uživatele buďto přes oddělení Service desk (za pomocí jednoho z nabízených rozhraní) nebo nový požadavek na službu schválený a zadaný zodpovědnou autoritou. Takto získaný požadavek je zadán do firemního systému, kde je kromě přidání jeho detailního popisu i vhodně kategorizován a je mu přiřazena priorita. Podle priority se jeho vyřešení buďto eskaluje okamžitě na vhodné oddělení nebo je zařazeno mezi ostatní položky čekající na zpracování podle pravidel procesu. Požadavek musí být schválen příslušným oddělením a ověřen. Kontrola slouží především pro přiřazení požadavku příslušné službě nebo skupině podpory. Zpracování požadavků probíhá podle modelu podobných požadavků, jenž jsou vedeny ve firemním systému. V případě, že žádný z modelů neodpovídá zadání, je požadavek předán ke zpracování určené skupině/oddělení. Pokud je pro splnění požadavku nutné provést určitou změnu, musí k tomu pověřený zaměstnanec oddělení Service desku zadat požadavek na změnu (RfC), jehož průběh zajišťuje proces Správa změn. V případě implementační změny je její nasazení do produkce a otestování řízeno procesem Správa releasů. Po kompletním dokončení všech nutných aktivit je záznam o požadavku na službu v systému uzavřen a uživateli je podána zpětná vazba. Popis kroků zobrazuje *(Obrázek 18)*.

Proces Plnění požadavků je spojen s procesy Správa přístupů a Správa financí, se kterými spolupracuje především ve fázi schválení a ověření.



Obrázek 18: Diagram procesu Plnění požadavků ve spojitosti s ostatními procesy (podle BPMN), zeleně vyznačená část označuje místo, kde proces z knihovny ITIL navazuje na metodiku Scrum

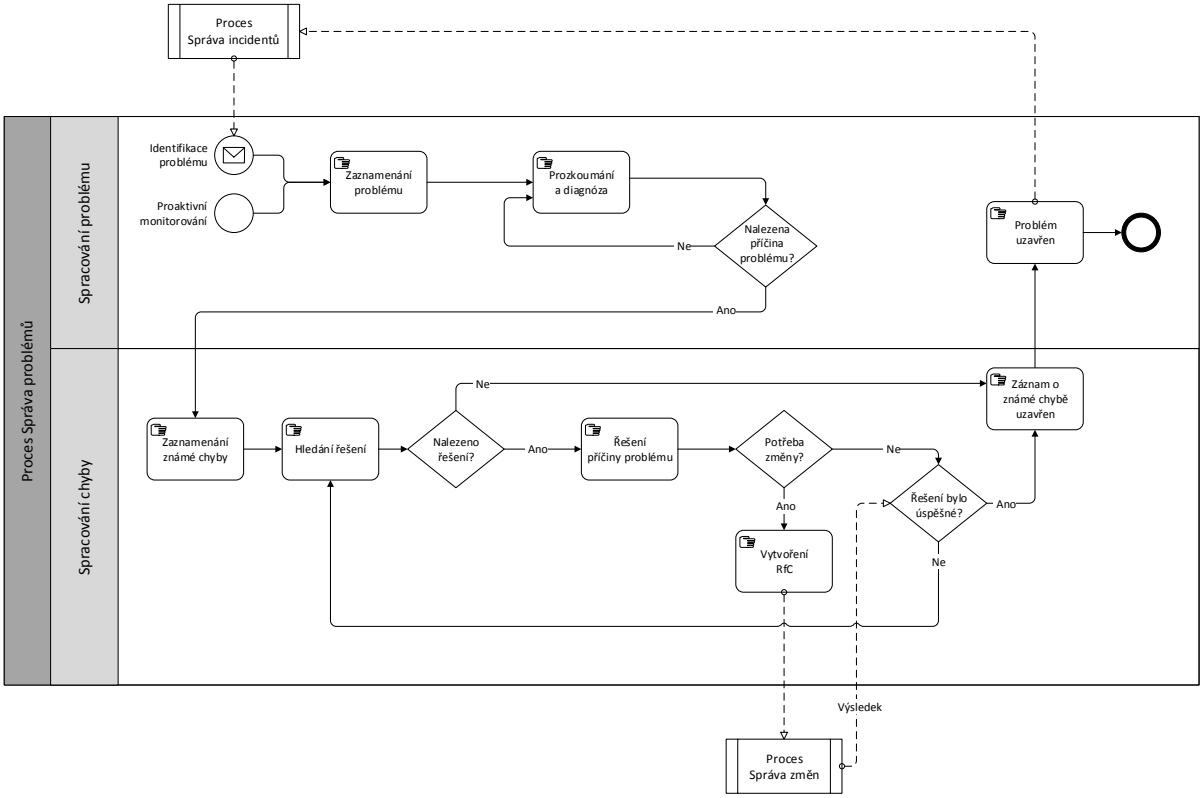
Proces Plnění požadavků je stejně jako proces Správa incidentů z větší části definován podle knihovny ITIL (první a druhá úroveň oddělení Service desku). Třetí úroveň zpracování požadavku je opět přímým vstupem do metodiky Scrum, která zajišťuje další zpracování.

8.3.4 Správa problémů

Vlastníkem procesu Správa problémů je Problem manager, který obvykle zastává i roli Team leadera v jednom z ostatních oddělení vývoje. Má na starost všechny aktivity týkající se řešení problémů, provádí pravidelné reporty výsledků řešení problémů a stanovuje metriky, podle kterých vyhodnocuje výkonnost a efektivnost procesu. Další podstatnou roli v procesu je Vlastník problému. Ten nese zodpovědnost za konkrétní problém, provází jej skrze celý jeho životní cyklus a má na starost jeho korektní uzavření spolu s popisem.

Vstupem procesu je problém, který lze rozeznat různými cestami. Buďto jako výstup ze Správy incidentů ve formě opakujících se incidentů, které jsou způsobené stejným problémem, nebo odhalením problému v některém z dalších oddělení (Service desk, Správa událostí) i díky proaktivní správě problémů. Nově rozeznaný problém je ihned zaznamenán do firemního systému pro záznam problémů, kde je kromě popisu i vhodně kategorizován. Následuje podrobná analýza, v rámci které se určí priorita problému, podle které se řešení neprodleně eskaluje nebo postupuje podle pravidel procesu. Cílem analýzy je stanovit příčinu vzniku problému a vytvořit záznam o známé chybě do databáze chyb, který bude sloužit v případě vzniku podobné chyby pro jejich rychlejší vyřešení. Následuje samotné řešení příčiny problému, která může zahrnovat implementaci změny v problémovém úseku (dále zpracovává

proces Správa změn). Pokud je po vyřešení problém úspěšně odstraněn, jeho záznam je v systému spolu s vysvětlujícími informacemi uzavřen spolu se záznamem známé chyby. Popis jednotlivých kroků zobrazuje *(Obrázek 19)*. Fungující a efektivní proces Správy problémů snižuje výskyty incidentů, čímž zvyšuje i spokojenost zákazníka. Jeho korektní implementace je tak důležitým doplňkem procesu Správa incidentů.



Obrázek 19: Diagram procesu Správa problémů ve spojitosti s ostatními procesy (podle BPMN)

Proces Správa problémů je definovaná kompletně podle metodiky ITIL, protože souvisí s procesem Správa incidentů a Správa změn, tedy neexistuje přímá vazba na metodiku Scrum. Navázání na vývoj podle Scrum je provedeno až podle již zmíněných procesů.

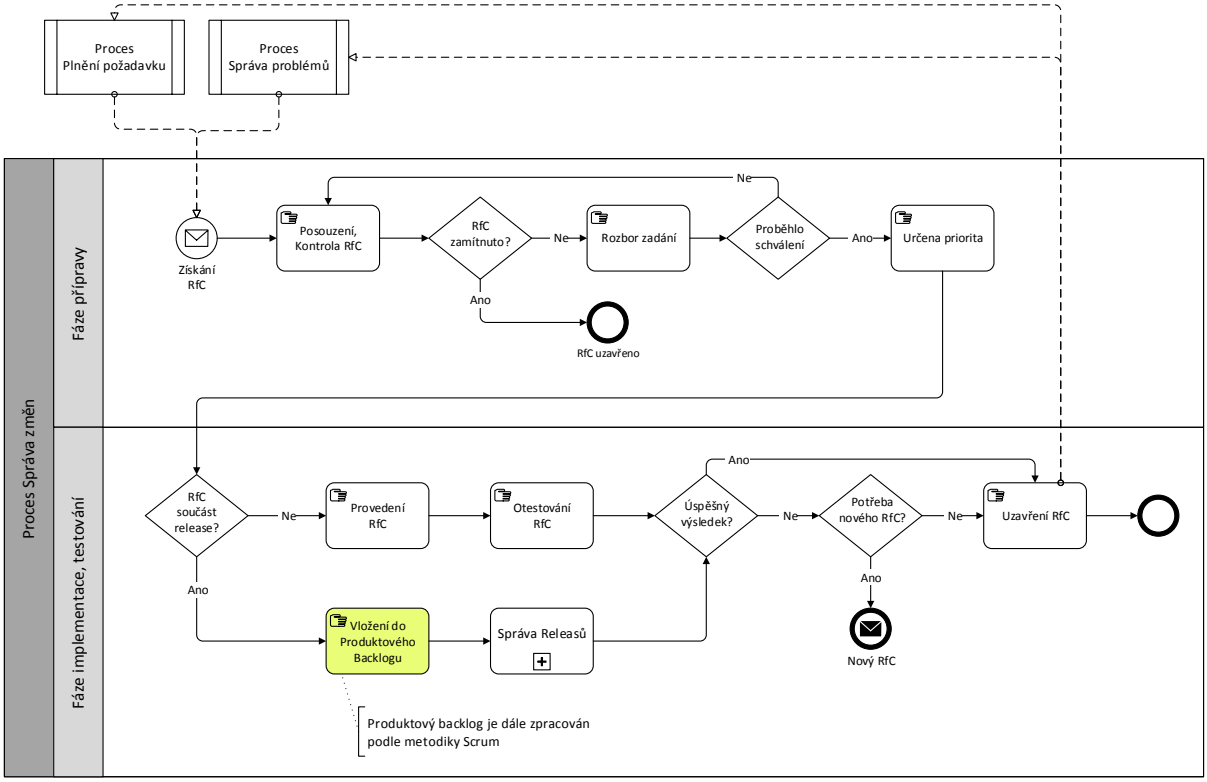
8.3.5 Správa změn

Vlastníkem procesu je Team leader oddělení, které má na starosti zpracování a řešení požadovaných změn. Vlastník procesu má na starost všechny aktivity týkající se změn a vytváří pravidelné reporty, díky kterým lze jednoznačně určit efektivitu a přínos procesu. Další důležitou rolí je vlastník zadané změny. Ten je zodpovědný za dodání detailního popisu a všech potřebných součástí a stopuje celý životní cyklus změny až po její uzavření. Role schvalovací autority má na starosti formální schválení žádosti na změnu.

Vstupem procesu je požadavek na změnu (RfC), který je zadán pro rozšíření/editaci funkcionality služby/systému nebo je předán z jiného procesu (Správa problémů, Plnění požadavků). Žadatel o změnu musí vyplnit všechny požadované informace (důvod změny, popis, závislosti, dopady, možná rizika, atd.) do firemního systému, který vygeneruje jednoznačné ID, podle kterého lze konkrétní žádost sledovat. Po zadání následuje posouzení každé žádosti o změnu a jejich kategorizace, v rámci které můžou být určité žádosti s odůvodněním i zamítnuty schvalovací autoritou. Po kontrole následuje

rozbor možných dopadů a potřebných zdrojů pro provedení změny, v rámci kterého je požadavku přiřazena priorita vypovídající o kritičnosti provedení změny a probíhá i plánování časové náročnosti provedení změny. Před samotnou implementací musí schvalovací autorita (jedna či více osob) formálně změnu schválit. Po implementaci následuje její otestování v podobě UAT (*„User Acceptance Test"*), které potvrzuje, že implementace změny odpovídá požadavkům zákazníka a je možné její řešení přesunout do produkčního prostředí. Samotná implementace už probíhá v rámci metodiky Scrum, tedy přesunem zadání požadavku na změnu do Produktového backlogu. Záznam o změně lze v systému uzavřít v případě, že otestování implementace po nasazení do produkce mělo pozitivní výsledek. Popis jednotlivých kroků zobrazuje *(Obrázek 20)*.

Požadavek na nový projekt je rovněž zpracován procesem Správa změn, na fázi schvalování a posouzení je kladen větší důraz a je během ní požadována větší kooperaci s businessem a získávání informací z procesu Správa strategie služeb IT a Správa portfolia služeb.



Obrázek 20: Diagram procesu Správy změn ve spojitosti s ostatními procesy (podle BPMN), zeleně vyznačená část označuje místo, kde proces z knihovny ITIL navazuje na metodiku Scrum.

Proces Správa změn je ovlivněn jak metodikou Scrum, tak knihovnou ITIL. Princip činností je definován podle ITIL, ale protože provedení většiny změn podléhá programovému zpracování, jsou aktivity upraveny tak, aby odpovídaly požadavkům Scrum. Po kategorizaci požadavku se dalších setkání a plánování účastní i vlastníci produktů, které spadají do určené kategorie. Přímé napojení na Scrum je skrze vložení požadavku na změnu do Produktového backlogu. Jednodušší přechod zajišťuje vhodná volba firemního systému.

8.4 Implementace

V předchozích kapitolách byly popsány typy požadavků, které se mohou dostat až do fáze implementace, tedy bude nutné jejich programové či systémové zpracování. Jednotlivé typy požadavků byly popsány společně s procesy, které řídí jejich životní cyklus a jsou popsány v procesní knihovně ITIL. Implementace různých druhů požadavků s různými prioritami a kategoriemi je nutné časově rozdělit mezi dostupné zdroje a zajistit jejich co nejefektivnější implementaci. Při větším množství je žádoucí pro aktivní správu požadavků použít vhodnou metodiku. V rámci popisované metodiky SCRIL bude tato část inspirovaná agilní metodikou Scrum, jejíž obecné principy byly popsány v (Kapitole 4).

Vstupem do fáze implementace jsou 4 typy požadavků: incident, žádost o změnu (RfC), požadavek na službu nebo nový projekt. Podle toho, o jaký typ požadavku se jedná, je požadavek předán na zpracování buď do Produktového backlogu nebo do Operativního backlogu. Zpracování požadavků se u obou backlogů zásadně liší, každý má svůj vlastní proces a pravidla. V této části jsou výstupy procesů z knihovny ITIL přesměrovány na vstupy zpracované metodikou Scrum, která bude dále popsaná. Činnosti v rámci metodiky jsou upraveny podle potřeb procesů z ITIL, např. rozdělením backlogů na Produktový a Operativní, čímž budou vyřešeny požadavky, které vyžadují rychlejší zpracování bez složitější přípravy. To, jakým způsobem mají být požadavky zpracovány knihovny ITIL neuvádí, proto je možné v tomto směru kompletně použít pravidel a výhod metodiky Scrum.

8.4.1 Operativní část vývoje

Operativní část vývoje se zabývá požadavky, které je většinou nutné vyřešit poměrně rychle, jejich vývoj není příliš složitý a nezabere mnoho času. Také platí, že vývoj těchto požadavků a jejich vydání do produkce není nebezpečný a nemá žádný podstatný dopad na business společnosti. S přihlédnutím na předchozí kapitoly, ve kterých byly popsány typy požadavků spolu s procesy, které je řídí, můžeme do operativní části zařadit tři potenciální typy požadavků: incident, požadavek na službu a požadavek na změnu (RfC). To ovšem neznamená, že tyto typy požadavků jsou zpracovány výhradně v operativní části, jak bude uvedeno dále.

O tom, jaký požadavek je vhodný pro vývoj operativní části se rozhoduje v prvních fázích jeho zpracování. V každém z jednotlivých procesů se tak děje při specifikaci zadání, určení priority a kategorizaci. Pro každý proces tuto zodpovědnost nese k tomu určený zaměstnanec daného oddělení a její důležitost by se neměla podceňovat. Pokud špatně zvolí kategorizaci a požadavek tak bude přiřazen špatné skupině vývojářů, spotřebují se zbytečně zdroje na analýzu požadavku, aby se nakonec zjistilo, že vlastně není určen jim. Správné určení priority je ještě podstatnější, především u potenciálního podcenění požadavku – v tom případě to může mít dopad i na spokojenost zákazníků, od kterých je získávána zpětná vazba.

Po určení priority a kategorie je požadavek předán do Operativního backlogu. Tam jsou všechny položky řazeny primárně podle kategorií, sekundárně podle priorit. Za každou kategorii nese zodpovědnost množina Scrum masterů, jejichž scrumové týmy do ní spadají. Scrum masteři pravidelně a kontinuálně přiřazují položky s nejvyšší prioritou konkrétním vývojářům v týmu, kteří nesou zodpovědnost za správné a úplné vyřešení daného požadavku v požadovaném čase. Samotné řešení požadavků z Operativního backlogu probíhá mimo metodiku Scrum, přesto by ji nemělo nijak narušovat. Každému vývojáři je z jeho celkového časového výkonu vymezen čas, který je určen řešení těchto typů požadavků. Ideálně by toto časové rozmezí mělo být mezi 20-40 % z celkového času s přihlédnutím na typ a množství dodávaných služeb ve společnosti. Během implementace jsou možné případné konzultace k řešení, jejich plánování a průběh ovšem nejsou součástí schůzek, které definuje

metodika Scrum. Kromě zmíněných konzultací v operativní části vývoje nejsou definovány žádné další povinné schůzky. Po implementaci probíhá testování vyvinutého požadavku v prostředí, ve kterém byl vyvíjen, ideálně jiným vývojářem než tím, který jej vyvinul. Po otestování je požadavek dále zpracován v rámci procesu Správa releasů (Kapitola 8.5).

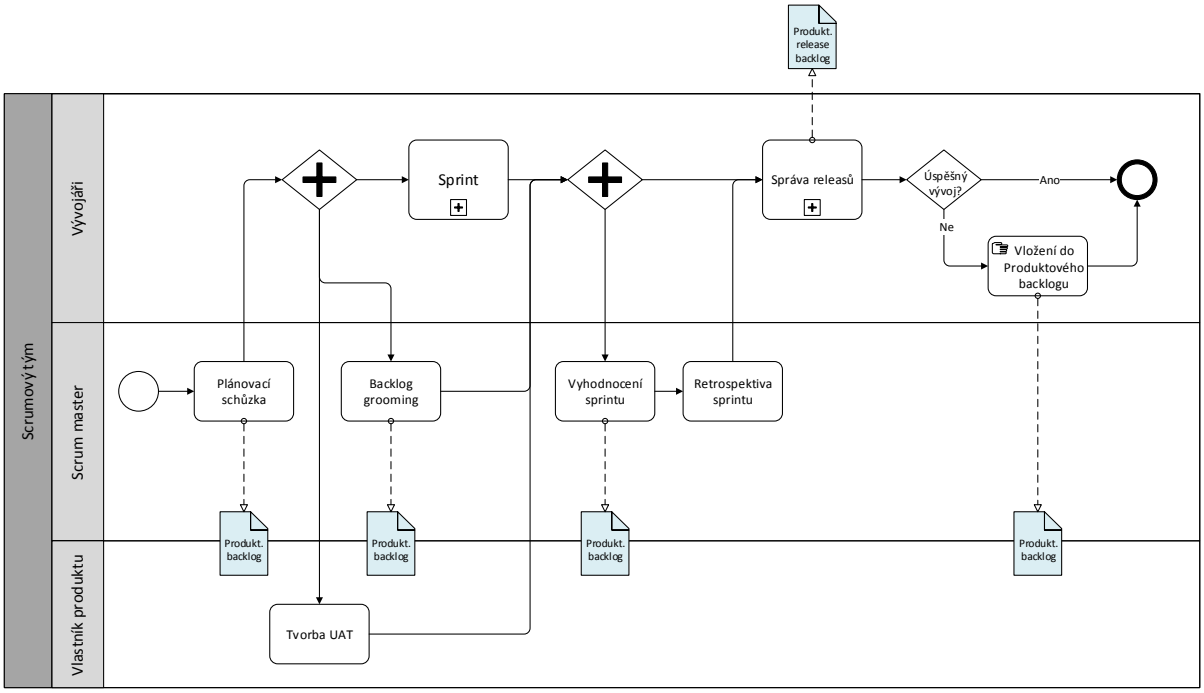
8.4.2 Produktová část vývoje

Produktová část vývoje se zabývá požadavky, které většinou nejdou vyřešit během kratší doby a jejich řešení vyžaduje delší plánování a přípravu. Jejich případné nevyřešení nebo špatné otestování navíc mohou mít vážné dopady na business společnosti. Do produktové části je primárně směřován jediný typ požadavků, a to požadavek na změnu (rozlišujeme jako požadavek na změnu a požadavek na nový projekt/produkt). Před samotným nasměrováním požadavků do Produktového backlogu, stejně jako u operativní části, i zde proběhne jejich zpracování v podobě určení priority a úvodní analýzy. V rámci kategorizace jsou přiřazeny ke konkrétnímu produktu a tedy i konkrétnímu Produktovému backlogu.

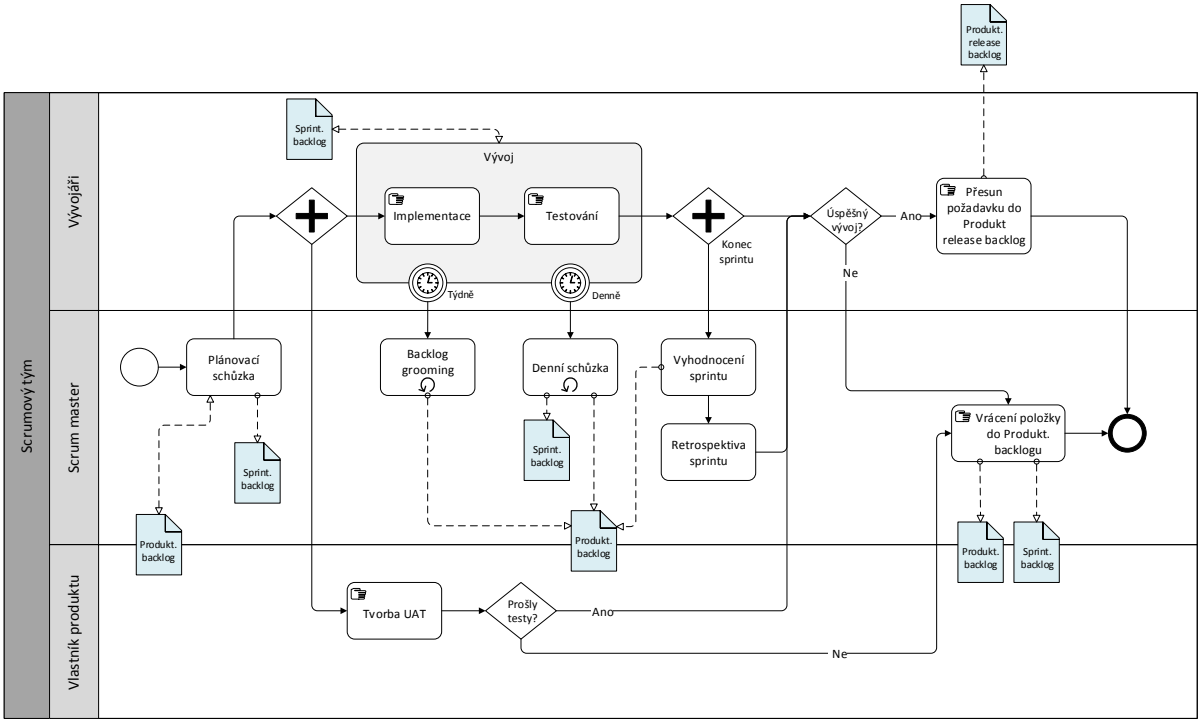
Produktový backlog je zpracováván cyklicky během sprintů. Délka sprintů je určena podle potřeb společnosti, ideálně by však neměla přesáhnout délku jednoho měsíce. Ihned po skončení jednoho sprintu následuje další, všechny mající stejnou délku. V rámci každého sprintu probíhá řada schůzek, na kterých se plánuje a kontroluje další vývoj zpracování požadavků v rámci projektu. Dodržování schůzek je důležité, protože právě schůzky v metodice Scrum pomáhají dodržovat její pravidla, čímž společnosti poskytují její hlavní výhody. Protože se definice schůzek metodiky SCRIL nijak zvlášť zásadně neliší od schůzek definovaných v metodice Scrum, nemá smysl je všechny rozepisovat znovu (detailní popis v Kapitole 4.4. V této části textu si jen velmi krátce shrneme základní skutečnosti pro rychlou rekapitulaci.

Denní schůzka probíhá denně v každém scrumovém týmu, ve stejný čas, ideálně by délka trvání neměla přesáhnout 15 minut. Je řízena Scrum masterem a kromě něj se ji povinně účastní všichni členové vývojového týmu. Můžou se jí účastnit i další lidé, nemají však právo schůzku jakkoli zdržovat ani během ní mluvit. Každý člen vývojového týmu během ní zodpovídá tři otázky: na čem právě pracuje, co udělal od poslední schůzky a co bude dělat do další schůzky.
Schůzka **Vyhodnocení sprintu** probíhá na konci každého sprintu, účastní se jí Scrum master, Vlastník produktu, vývojový tým a všichni ostatní, kteří jsou spojeni s daným produktem. Probírají se udělané věci během posledního sprintu, na základě čehož se aktualizuje Produktový backlog. Po ní probíhá **Retrospektiva sprintu**, které se účastní Scrum master s vývojovým týmem a probírají uplynulý sprint pro zhodnocení použitých procesů, postupů a nástrojů s cílem navrhnout zlepšení pro další sprint. Většinou hned po těchto dvou schůzkách následuje další, **Plánovací schůzka**, která probíhá těsně před začátkem dalšího sprintu a účastní se jí Vlastník produktu, Scrum master a další důležití lidé spojení s vývojem produktu. Z Produktového backlogu se vybírají položky, které jsou přesunuty do Backlogu sprintu, tzn. je naplánovaná jejich implementace během příštího sprintu. Další užitečnou schůzkou je **Backlog grooming**, které se účastní Vlastník produktu spolu se Scrum masterem a celým scrumovým týmem. Během ní společně prochází jednotlivé položky produktového backlogu, diskutují jejich obsah a rozkládají je na menší části, které lze jednodušeji zpracovat. Výhoda této schůzky je před příprava Produktového backlogu před plánovací schůzkou, jelikož tým má již přehled o práci, která bude potřeba vykonat v dalším sprintu. Měla by se konat každý týden a její délka by měla být úměrná délce ostatních schůzek.
Pro větší společnosti s většími projekty je vhodné do plánu schůzek zařadit i schůzku **Scrum scrumů.** Je to schůzka, na které členové různých scrumových týmů různých produktů diskutují o své práci, především o těch částech software, které se překrývají s jinými a jejich integrace do celku. Z každého týmu je na schůzku vyslán jeden vývojář nebo Scrum master, který reprezentuje potřeby celého týmu.

Schůzka by se ideálně měla konat každý týden, délka závisí na množství věcí, které týmy mají k prodiskutování, neměla by však trvat příliš dlouho na to, aby zdržovala zúčastněné od produktivní práce. Hlavním přínosem schůzky je podpoření komunikace mezi týmy.



Obrázek 21: Produktová část vývoje požadavků metodikou Scrum (podle BPMN)



Obrázek 22: Produktová část vývoje požadavků metodikou Scrum zobrazující průběh jednoho sprintu (podle BPMN)

Během sprintu, provázeným řadou schůzek, probíhá samotná implementace požadavků a zpracovávání UAT výsledku Vlastníkem produktu. Na implementaci včetně následného otestování vyvíjených položek by každý vývojář měl mít vyhrazeno 50-70% z celkového času (operativní vývoj 20-40%, zbylý čas je věnován vzdělávání a administrativním povinnostem). Po implementaci probíhá testování položky na vyvíjecím prostředí. Tu by měl ideálně otestovat vývojář, který ji nevyvíjel. Po otestování je požadavek předán procesu Správa releasů, který zajišťuje jeho převod a otestování na testovacích prostředích a konečné nasazení na produkční prostředí. *(Obrázek 21)* zobrazuje aktivity ve scrumovém týmu, které popisuje metodika Scrum, *(Obrázek 22)* pak zobrazuje aktivity ve scrumovém týmu v rámci jednoho sprintu.

8.5 Testování a nasazení

Po implementaci a otestování všech typů požadavků na vývojovém prostředí následuje jejich další zpracování, které zajišťuje proces Správa releasů. Vlastníkem procesu je Release manager, který má na starost veškeré aktivity, které jsou v rámci procesu prováděny. Kromě toho stanovuje metriky, podle kterých se hodnotí efektivita a výkonnost procesu a pravidelně provádí reporty, které vyhodnocují výsledky procesu. Provádění reportů je v této fázi vývoje velmi důležité, protože právě oddělení releasů uzavírá celý vývoj různých druhů požadavků, které skrze Service desk i scrumové týmy nasazuje na produkční prostředí. Má tedy všechny potřebné informace k možnosti zhodnocení úspěšnosti vyvíjení zadaných požadavků. Reporty by měly být vytvářeny pravidelně, ideálně měsíčně/čtvrtletně podle potřeb společnosti, dostatečně specializovaným pracovníkem.

Vstupem procesu je množina implementovaných požadavků z operativní nebo produktové části vývoje, otestovaná na vývojovém prostředí. Požadavky jsou shromažďovány a rozdělovány podle toho, kterým vývojem se do procesu dostaly, protože předpoklad požadavků vydání produktových i operativních požadavků se v zásadě liší. U operativních položek se předpokládá, že jejich vydání je vyžadováno poměrně rychle a protože jejich implementace není příliš náročná, větší testování není nutné. Oproti tomu vydání produktových položek je nutno naplánovat z důvodu rozsáhlejších projektů a požadavků, které musí být dobře otestované i na testovacích prostředích. Datum nasazení do produkce je dopředu domluvené a plánované.
V této části se tedy výstupy z metodiky Scrum v podobě naimplementovaných a otestovaných požadavků, stávají vstupem do procesu Správa releasů, jehož aktivity jsou převzaty z knihovny ITIL. Ovlivnění metodikou Scrum je v tomto procesu značné, protože veškeré vydávání musí být vhodně přizpůsobeno výstupům ze Scrum, především časové plánování. Zároveň ale musí být dodrženy všechny důležité aktivity pro release. Proces je dále podle Scrum přizpůsoben např. rozdělením releasů podle toho, z jakého typu backlogu jsou přebírány požadavky. Každý typ releasů tak podléhá upraveným pravidlům podle potřeb.

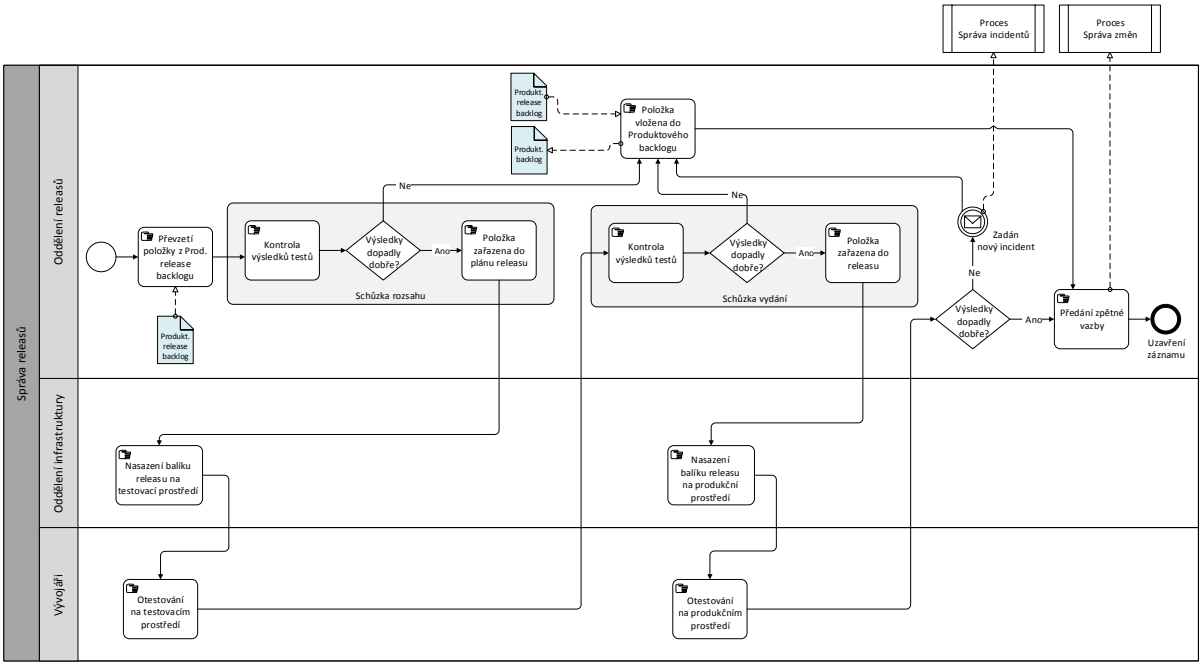
8.5.1 Operativní položky

Požadavky, které byly vyvinuty v operativní části vývoje (mimo pravidla metodiky Scrum), jsou předány Správě releasů a směřovány do operativních releasů. Ty jsou v kalendáři releasů naplánovány na pravidelné dny – jednou či dvakrát týdně podle počtu vyvíjených požadavků. Hotové operativní položky jsou shromažďovány do Operativního release backlogu spolu s údaji o technických krocích pro release. Předpokládá se, že pokud byla operativní položka zařazena do Operativního release backlogu, její testy na vývojovém prostředí dopadly dobře, proto se primárně nepožaduje žádný dokument, který by to dokládal (oproštění od zbytečných dokumentací). Po vybrání položek pro

konkrétní operativní release se zaevidují všechny technické požadavky na release a vytvoří se z nich plán, podle kterého Release manager postupuje v den release ve spolupráci s infrastrukturou společnosti. Nasazení položek (mimo výjimky) probíhá přímo do produkčního prostředí, ve kterém jsou následně provedeny testy. Výsledky těchto testů stanoví, jestli implementace požadavku byla úspěšná, či nikoliv.

8.5.2 Produktové položky

Požadavky, které byly vyvinuté v produktové části vývoje za použití metodiky Scrum, jsou předány Správě releasů a směrovány do produktových releasů. Data produktových releasů jsou plánována dopředu tak, aby byla synchronní se sprinty – za tímto účelem může být přítomnost Release managera na plánovacích schůzkách velmi přínosná. Naimplementované a na vývojovém prostředí otestované požadavky jsou směrovány do Produktového release backlogu, odkud jsou převzaty Release managerem. Je nutné mít výsledky testů od všech přijatých položek.
Před konečným zařazením položek do release je konaná **schůzka rozsahu**, na které se kromě Release managera sejdou všichni Scrum masteři dodaných položek, Vlastníci daných položek, zástupce infrastruktury a další důležití lidé spojení s dodávanými položkami. Release manager diskutuje se všemi přítomnými zařazení položek do releasů, ověřuje jejich funkčnost na vývojovém prostředí, ověřuje výsledky UAT a prochází seznam potřebných technických kroků pro jejich vydání. Výhodou a nutností pořádání této schůzky je mimo jiné to, že se definitivně zkontrolují všechny výsledky z vývoje a vyjasní se, které položky v dané datu mohou a které nemohou být vydány, navíc si infrastruktura udělá přehled o tom, kolik zdrojů bude nutné pro přípravu vydání.



Obrázek 23: Diagram procesu Správa releasů ve spojitosti s ostatními procesy (podle BPMN)

Po schůzce se z připravených položek vytvoří balík, který je infrastrukturou nasazen na testovací prostředí. Ihned po nasazení scrumové týmy testují jejich funkčnost, oddělení releasů chystá detailní plán kroků vydání v den release. Dva dny před plánovaným datem vydání probíhá **schůzka vydání**, kterou vede Release manager za účasti Scrum masterů, Vlastníku položek, člena infrastruktury a dalších. Diskutují se výsledky testů z testovacích prostředí, na základě kterých se naposled upraví

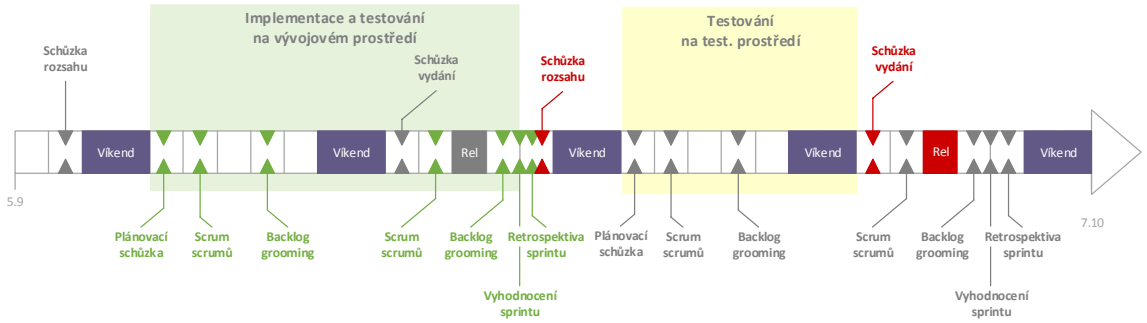
seznam položek, které budou vydány do provozního prostředí. Zamítnuté položky se vrací do Produktového backlogu k dalšímu zpracování. Na schůzce Release manager prochází plán vydání položek a ujišťuje se, že všichni rozumí všem krokům a souhlasí s nimi. Poslední den před vydáním je vyhrazen na vyřešení potenciálních problémů a nejasností s cílem je minimalizovat.

V den vydávání se postupuje podle stanoveného plánu, všichni vývojáři z týmů, jejichž položky jsou součásti release jsou ihned po vydání připravení znovu otestovat jejich funkčnost a oddělení správa releasů dodat záznam o výsledku testování. Při zjištění nefunkčnosti některé z položek release je zjištěná chyba do firemního systému zadána jako nový incident a celý cyklus se tak opakuje. V opačném případě je záznam o požadavku v systému uzavřen a zákazníkovi je v případě nutnosti podána zpětná vazba o vyřešení jeho požadavku. Všechny aktivity v propojení s ostatními procesy zobrazuje *(Obrázek 23)*.

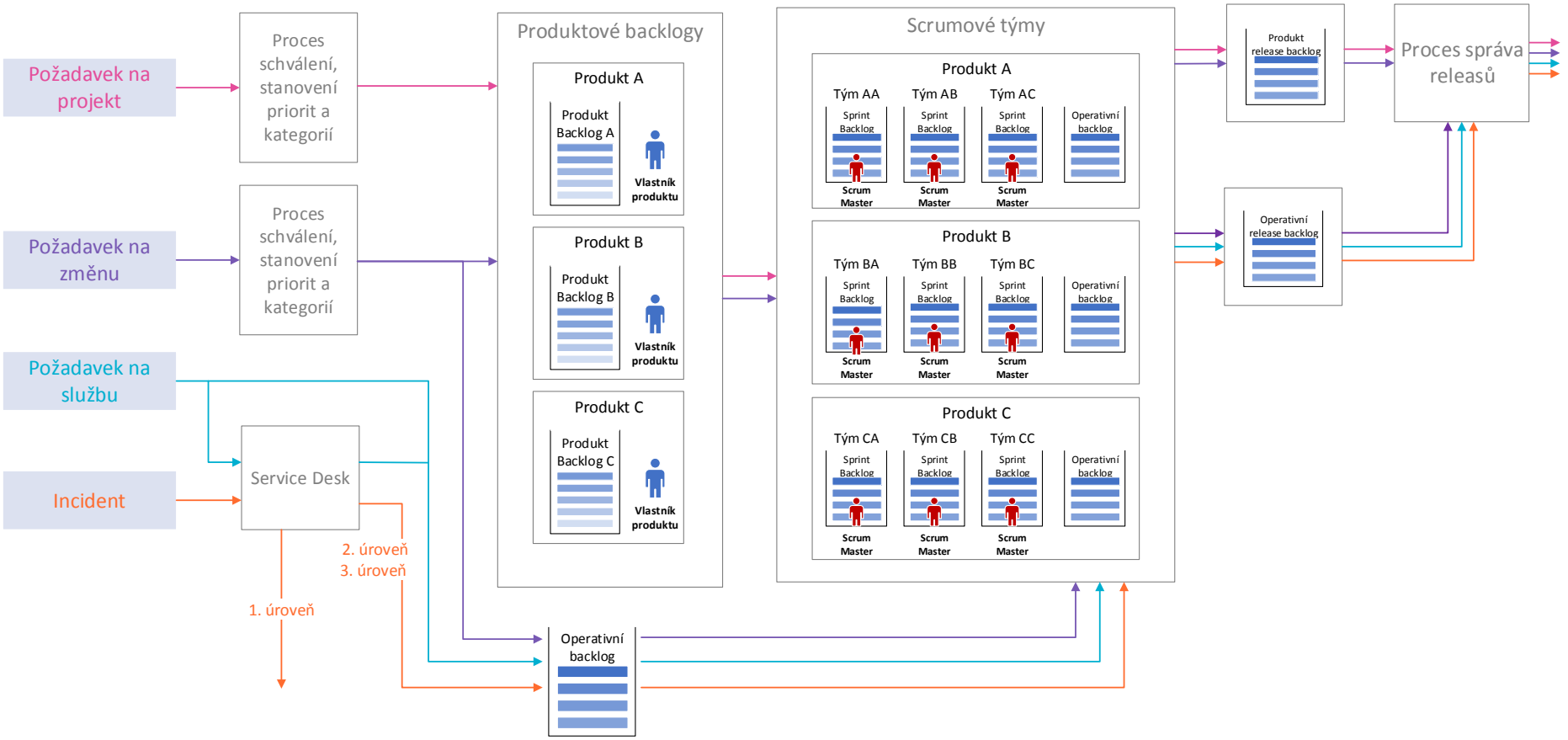
8.6 Shrnutí

Pro shrnutí metodiky SCRIL, neboli propojení vybraných procesů s aktivitami metodiky Scrum, jsou zde přiloženy dva obrázky. *(Obrázek 24)* zobrazuje časovou osu, která obsahuje schůzky a aktivity v rámci jednoho sprintu, včetně následného vydání do produkce. *(Obrázek 25)* zobrazuje různé typy požadavků (každý zpracován jiným procesem knihovny ITIL) jako vstup do vývoje pomocí metodiky Scrum. Výstup z metodiky Scrum je předán jako vstup do procesu Správa releasů, který ukončí celý životní cyklus požadavků.

U každé popsané fáze bylo řečeno, jestli procesy v ní jsou přejaty spíše z knihovny ITIL nebo metodiky Scrum a jak se vzájemně ovlivňují. Fáze zpracování požadavků obsahuje procesy hlavně podle knihovny ITIL, především jejich výstup v podobě předání k implementaci metodice Scrum musí být vhodně přizpůsoben tak, aby se dal lehce použít ke zpracování (vhodný firemní systém pro správu požadavků, různé typy backlogů apod.). Metodiky Scrum vyvíjí požadavky podle vlastních pravidel, které až na rozdělení backlogů na operativní požadavky a produktové požadavky, odpovídá pravidlům Scrum. Výstup je předán dalšímu procesu z knihovny ITIL – Správa požadavků, jehož aktivity jsou upraveny tak, aby odpovídaly výstupům z metodiky Scrum. Např. rozdělení typů release podle toho, z jakého backlogu je požadavek získán nebo přizpůsobení časovému plánu vydávání release do produkce.



Obrázek 24: Časová osa aktivit a schůzek během sprintu s přechodem do Správy releasů, kde zeleně vybarvené schůzky zobrazují aktuální sprint, červeně vybarvené schůzky release pro aktuální sprint a šedě vybarvené schůzky označují minulý resp. nadcházející sprint.



Obrázek 25: Tok různých typů požadavků skrze procesy knihovny ITIL a metodiky Scrum

9 Aplikace pro podporu reportingu

Jako součást zadání svojí práce jsem měla navrhnout a implementovat takovou aplikaci, která podpoří navrhovanou metodiku SCRIL. Metodika SCRIL sestává ze spojení agilní metodiky Scrum a procesní knihovny ITIL. Tok procesů mezi jednotlivými metodikami byl navržen v předchozí kapitole, je nutné se však zmínit i o fyzickém spojení obou metodik.

Většina společností pro správu požadavků v rámci společnosti využívá určitý systém, který jí to podstatně ulehčí. Do systému jsou zadávány různé typy požadavků (ať manuálně, či automaticky), které mají být vyřešeny. Tyto typy systémů většinou (podle mých zkušeností) společnosti zakupují, jelikož jsou natolik náročné na požadavky, že jejich implementace a údržba by se společnosti finančně nevyplatila. V případě existence oddělení Service desku je takový systém přímo nezbytný – pro příklad takových systémů bych ráda uvedla ty, se kterými jsem měla šanci se setkat: iSupport, Service Now. U těchto dvou zmíněných systémů je jejich podpora zaměřena především na zadávání a sledování požadavků takových typů, které nevyžadují složitější přesuny stavů, vzájemné propojování podle závislosti apod. Zadané požadavky, které vyžadují implementaci a zahrnutí či propojení s dalšími požadavky, jsou přesouvány do dalších, pro to přizpůsobených systémů. Jako příklad takového systému bych ráda uvedla Bugzilla nebo Atlassian JIRA.

Druhý zmíněný systém – Atlassian JIRA používá i společnost AVG Technologies pro správu a sledování projektů. Pro každý projekt tam lze vytvořit vlastní oddíl, v rámci kterého se spravují s projektem společné činnosti, zadávají se úkoly, přiřazují se konkrétním lidem apod. Jako samostatný projekt lze považovat i proces metodiky Scrum, ke kterému tam lze vytvořit vlastní Scrum board pro každý scrumový tým a v rámci něj sledovat postupy a přesuny požadavků v rámci týmu, projektu, backlogu i jejich stavu. Stejně tak pro oddělení Správu incidentů lze založit vlastní projekt, který bude spravovat incidenty, obdobně pro Správu problémů. Oddělení Správy releasů je na tomto systému zcela závislé, protože na základě požadavků v něm zadaných plánuje a provádí vydávání releasů. Systém obsahuje mnoho pluginů a součástí, které lze poměrně slušně přizpůsobit požadovaným úkonům a účelům. Důležitou skutečností je, že právě ze systému JIRA Team leadeři a manažeři získávají informace o tom, jak který projekt byl úspěšný a jestli byly splněny naplánované výsledky. Informace lze z JIRA vyexportovat do různých formátů: xml, xls, doc; se kterými lze dále pracovat. Nejčastěji se provede export dat podle zadaného vyhledávacího filtru do excelovského formátu, ve kterém se ručně nastaví vykreslení požadovaných grafů. Ty slouží pro jasné zobrazení výsledků do pravidelných reportů nebo při prezentacích výsledků jednotlivých oddělení pro vyšší vedení.

9.1 Proč reporting?

Na konci kapitoly 8: Návrhu metodiky SCRIL jsem se zmínila o tom, jak moc je důležité sledovat úspěšnost zavedených procesů a stanovit si pro to vhodné metriky. Proces, který je dobře a podrobně popsán může být po určitou dobu pro společnost přínosný, pokud se však nebude testovat jeho účinnost, může se stát neefektivní a ohrozit i výstup pro zákazníka a fungování celé firmy. Z tohoto důvodu považuji tuto část návrhu/správu procesů za velmi důležitou a v případě zavedení moji metodiky do praxe zcela nezbytnou.

Když se znovu podíváme na navrženou metodiku SCRIL, zjistíme, že posledním článkem celého vývojového procesu jednotlivých požadavků je proces Správa releasů, přes který by měly být všechny požadavky přesouvány z vývoje do produkce. A právě v tomto bodě je nejlépe možné zhodnotit

úspěšnost metodiky, která spojuje ITIL a Scrum. Pokud tuto myšlenku převedeme do praxe v kontextu AVG Technologies, musíme se zaměřit na systém JIRA, ve kterém jsou zadávány a sledovány veškeré požadavky, které skrze Správu releasů probíhají. Momentálně společnost nemá k dispozici vhodný doplněk, který by automaticky generoval grafy podle dotazu přímo ze systému JIRA, proto to týmoví vedoucí provádí pomocí excelovských tabulek a grafů z nich. Právě tohoto nedostatku jsem se rozhodla využít a navrhnout aplikaci, která automaticky ze systému JIRA po zvolení filtru vykreslí daný typ grafu s požadovanými údaji v potřebném časovém intervalu.

9.1.1 Komu je aplikace určena

Protože je aplikace napojena přímo na systém JIRA pro získání výsledků z projektu sloužící pro proces Správu releasů, je předně určena k používání hlavně v tomto oddělení. Nutno zmínit, že jedním z výstupů tohoto procesu je zhodnotit předešlé aktivity a poskytnou přehled o tom, kolik releasů společnost během roku provádí a s jakým výsledkem. Rovněž bude možné z výsledků vyčíst a zhodnotit, jak úspěšná byla integrace obou metodik, což je hlavním tématem celé této práce. Konkrétní využití s příklady bude uvedeno v dalších kapitolách.

Kromě oddělení Správy releasů je aplikace určena i pro všechny ostatní oddělení, která jsou jakkoliv spojena s dodáváním a vývojem požadavků do metodiky SCRIL včetně Scrum masterů, kteří díky aplikaci mohou sledovat efektivitu jednotlivých členů scrumových týmů a vyvozovat z toho výsledky a návrhy pro potřené změny. Příklady pro jednotlivé procesy budou opět uvedeny v dalších kapitolách.

9.2 Požadavky na funkcionalitu

Jelikož předpokládám, že aplikace bude využívána i v praxi (minimálně v rámci AVG Technologies) a je za tímto účelem od začátku vyvíjena, bylo nutné požadavky na ni konzultovat s příslušnými vedoucími a lidmi, kteří by ji potenciálně mohli využívat. Po konzultacích jsem stanovila základní požadavky, podle kterých jsem přizpůsobila návrh.

Hlavním požadavkem na aplikaci je, aby její použití bylo možné bez předchozích proškolení a bylo maximálně intuitivní – právě na tento požadavek jsem se nejvíce zaměřila. Aby toho bylo možné dosáhnout, musel být návrh ovládání aplikace co nejjednodušší a oproštěn od zbytečných možností výběru ideálně tak automatickými úpravami jednotlivých filtrů.

Dalším z požadavků bylo zaintegrování aplikace do jedné z existujících firemních stránek, které slouží podobným účelům a to ze dvou důvodů. Do systému JIRA, ze kterého jsou získávány data pro další zpracování, mají přístup pomocí přihlašovacího jména a hesla pouze zaměstnanci firmy. Každý z nich má dále ke svému účtu přiděleny uživatelská práva, která mimo jiné dovolují zobrazit a pracovat s vybranými projekty. Abychom aplikaci odprostili od kontroly hesla, jména a pravomocí k prohlížení, přidáme funkcionalitu do další z firemních aplikací, ke kterým mají přístup pouze uživatelé připojení na firemní síť. Navíc spojení aplikací a jejich spuštění ze stejného místa usnadní orientaci uživatelů a zkrátí čas hledání samotné aplikace mezi obrovským množstvím firemních dat, informací a aplikací. Tento důvod považuji za podstatný, protože přímo podporuje výhody, které jsou poskytnuty zavedením agilních metodiky – strávit minimum času hledáním a děláním nepodstatných věcí a soustředit se na vývoj produktu.

Další požadavky se vztahují přímo na funkcionalitu aplikace. Bylo nutné v jednotlivých odděleních sehnat informace o tom, jaké grafy nejčastěji pro zobrazení reportů používají a které informace v nich zobrazují. Podle toho byl přizpůsoben výběr omezujících podmínek pro filtry tak, aby kromě nich

neobsahoval žádné další nepožadované možnosti (čímž zkrátí čas před vykreslením grafu), ale zároveň nebylo obtížné s přicházejícími novými požadavky další možnosti doplnit.

Jak již bylo řečeno, použití aplikace se předpokládá především v rámci procesu Správa releasů. V rámci něj se každý měsíc vytváří několik typů grafů, které nejlépe ukazují účinnost plnění stanovených metrik. Grafy, které se tak každý měsíc vytvářejí, zůstávají stejné, pouze se posouvá časový interval zobrazovaných výsledků. Proto je víc než vhodné do aplikace zabudovat funkcionalitu, která dovolí použít předvyplněnou šablonu, u které bude stačit změnit data intervalu a ušetřit čas.

9.3 Návrh aplikace

Jeden z požadavků na aplikaci byla její integrace do již existující aplikace společnosti AVG Technologies. Pro tento účel jsem po konzultaci s Release managerem využila interně naprogramovanou webovou stránku, která slouží pro jednodušší orientaci procesem Správy releasů. Na této stránce mohou uživatelé vidět velmi pěkně graficky přehled měsíčně plánovaných releasů a pro zvolený konkrétní release rozpis jednotlivých položek v něm obsažených s cílem kontroly všech požadovaných součástí ke každé položce. Kromě toho umožňuje Release managerovi tvorbu časových os, které pro každý release zobrazují důležité milníky, podle kterých se orientují vývojové týmy a další osoby, které jsou jakkoliv spojeny s vydáváním releasů.

9.3.1 Základní návrh

Tato webová aplikace je naprogramovaná pomocí jazyků php, html, css a javascript. Návrh její architektury odpovídá návrhovému vzoru alias architektuře „MVC“ (Model-View-Controller), která se skládá ze tří hlavních součástí:

- Pohled (view) je část, která převádí data reprezentovaná v Modelu do podoby vhodné pro uživatele, od kterého přímo získává další informace. Aplikace pro tento účel používá šablonovací systém „*Smarty Template Engine*“, který je napsaný v jazyce php a je volně stažitelný k používání. Jeho výhodou je právě oddělení funkcionální části od zobrazovací. Název šablon vytvořené pomocí Smarty koresponduje podle názvu Kontrolerů, s koncovkou „`.tpl`“.
- Model je část architektury, která obsahuje hlavní výpočetní logiku celé aplikace, reprezentuje zdroj informací, se kterými aplikace pracuje.
- Kontroler (controller) funguje jako řadič, který reaguje na změny a požadavky od uživatele a kontaktuje Model pro zajištění aktuálních informací k zobrazení pro Pohled.

Celá aplikace používá jako základní stavební kámen objektově orientovaný webový aplikační framework Zend, který je na internetu dostupný volně ke stažení a je implementován v jazyce php. Použitím jeho modulů lze docílit fungování právě podle architektury MVC.

9.3.2 Návrh pro reporting

Samotná část funkcionality v rámci reportingu byla rozvrhnuta do 3 základních částí: nový graf, nová šablona a použití šablony pro časový interval. Pro každou část v rámci architektury MVC je určen vlastní Pohled pro získání informací od uživatele (zdrojová složka `/tpl` soubory `new-report.tpl`, `template-new.tpl` a `template-report.tpl`). Popis jejich funkcionality přímo souvisí s používáním celé aplikace a bude rozebrán v další kapitole. Se všemi Pohledy spolupracuje skript

`reports.js`, který umožňuje dynamicky měnit vzhled formuláře podle zvolených parametrů od uživatele.
Jako řadič pro zpracování všech Pohledů je určena třída `ReportsController.php`, která obsahuje stěžejní funkce pro každý z pohledů: `newReportAction()`, `templateNewAction()` a `templateReportAction()` atd.; a podle požadavků od uživatele postupně kontaktuje jednotlivé modely.

Modely jsou logicky rozděleny do několika tříd:

- `Reports.php` – třída obsahuje hlavní aplikační logiku. Pro získané položky z databáze (funkce `getIssues()`) jsou zde funkce pro jejich porovnávání, získávání různých typů údajů, filtrování a zpracování výsledných dat pro vykreslení do grafu.
- `ReportsCharts.php` – třída určena pro práci s uloženými grafy pro šablony v databázi. Parametry grafů jsou uloženy v tabulce s grafy, každý s unikátním názvem a samotnými parametry opět jako textem v jediném sloupci.
- `ReportsTemplates.php` – třída určena pro práci s uloženými šablonami v databázi. Jednotlivé šablony jsou v tabulce uloženy s unikátním názvem (*id* jako primární klíč) a v jediném sloupci je uložen jako text výčet grafů v šabloně obsažených.
- `Chart.php` – třída uchovávající důležité parametry ke každému grafu uloženému v šabloně v databázi.

Po získání dat pro grafy ve vhodném formátu přichází na řadu jejich vykreslení. K tomuto účelu byly využity interaktivní grafy Highcharts [5]. Pro studijní účely je licence zdarma, v případě odsouhlasení po předvedení aplikace bude muset společnost zakoupit licenci k používání pro firmy s finančním výdělkem. Grafy jsou implementované v jazyce JavaScript, pro jejich správné použití je nutné do zdrojových kódů importovat knihovnu `jquery.js` a skript `highcharts.js`, který obsahuje zdrojové kódy pro vykreslování grafů. Grafy jsou dostupné v mnoha různých typech, pro použití určitého z nich je třeba do kódu vložit předdefinovaný úsek kódu grafu, kterému je třeba předat zdrojová data pro vykreslení. Vykreslování grafů probíhá ve všech případech v části Pohled (např. `chart.tpl`).

9.4 Použití

Aplikace na podporu reportingu je spustitelná na počítači s nainstalovaným webovým prohlížečem (omezení vzhledem ke konfiguraci počítače jsou dány právě možností spuštění webového prohlížeče) a zapnutým JavaScriptem. Vzhledem k tomu, že se jedná o firemní aplikaci, která obsahuje důvěrné data, je možné ji spustit až po přihlášení do firemní sítě.

V této kapitole si popíšeme, jakým způsobem se aplikace pro generování reportů používá. V zásadě ji lze logicky rozčlenit na tři hlavní části, na které se lze výběrem dostat z webové stránky rm.cz.avg.com/reports (stránku ukazuje (*Obrázek 32*) Příloha A) – Nový graf, Nová šablona a Použití šablony.

9.4.1 Nový graf

V této části lze vytvořit nový, nezávislý graf navolením vhodných parametrů. Z hlavní stránky se na ni dá dostat zvolením možnosti „*New Chart*“. Zadávání parametrů grafu je rozděleno na dvě části. V první části je uživatel vyzván k zadání tří hlavních údajů: datum od, datum do a typ požadavku. Pro typ

požadavku (angl. Issue type) je na výběr ze dvou možností – Release a Release Item. Typ release označuje všechny release, které se vydávaly a typ release item jsou všechny jednotlivé položky, které jsou zapouzdřeny v release a vydávají se společně v rámci jednoho release. Po navolení těchto tří parametrů se automaticky pošle dotaz do databáze, který získá data. V další fáze je důležité zvolit typ grafu, který si uživatel přeje vykreslit (prozatím možnost výběru z koláčového grafu, sloupcového grafu, liniového grafu a kombinace všech zmíněných) a jakou hodnotu chce, aby graf reflektoval (např. když chci vykreslit všechny release v uplynulém měsíci, jako hodnotu pro vykreslení zvolím „Release Type"). Následuje možnost přefiltrovat data podle doplňujících parametrů tak, aby se daly vykreslit i složitější dotazy. Možnosti výběru parametrů pro filtry jsou řešeny formou checkboxů, kdy se po zaškrtnutí objeví okno s možnostmi výběru pro daný parametr (zobrazuje (*Obrázek 33*) Příloha A). Potvrzení výběru pomocí tlačítka „Plot Chart" vykreslí graf. V případě zvolení takových omezujících podmínek pro filtr, které nevrací žádná data, se na obrazovce objeví upozornění.

(Pozn.: pro správné použití aplikace se předpokládá, že je uživatel seznámen s názvoslovím a významem jednotlivých parametrů obsažených v systému JIRA v projektu Release Items).

Tato funkcionalita je určena lidem, kteří potřebují rychle a snadno zjistit stav fungování nějakého procesu nebo činnosti, aniž by strávili delší dobu získáváním a grafickým upravováním získaných informací. V rámci metodiky SCRIL bude doporučena k používání především Team Leaderům jednotlivých oddělení a Scrum masterům, obecně všem těm, kteří vykazují jakékoliv výstupy ve formě zhodnocení klíčových ukazatelů výkonnosti procesů. U vytváření jednorázových grafů se dá předpokládat, že půjde spíše o informace operativní povahy.

Příklad 1.: Scrum master týmu A potřebuje zjistit, kolik incidentů nejvyšší priority se podařilo opravit a vydat ve spolupráci s oddělením Správy releasů jeho scrumovému týmu během posledního týdne.

Příklad 2.: Release manager potřebuje zjistit, kolik požadavků bylo vydáno z Operativního release backlogu během posledních dvou týdnů.

9.4.2 Nová šablona a použití šablony

Pro přidání nové šablony je nutné zvolit z hlavní stránky možnost „*Add to Template*", poté se otevře stránka s formulářem pro zadání vstupních informací o šabloně. Nejprve je nutné vybrat, jestli chceme graf přidat do již existující šablony nebo vytváříme šablonu novou – v tom případě je nutné zadat i jméno šablony do připraveného formuláře. Poté stačí navolit parametry pro vykreslení (stejný princip jako u tvoření nového grafu, viz předchozí kapitola) a stisknout tlačítko „*Save*", které zadané informace uloží do databáze. Jednotlivé grafy se do šablony přidávají postupně.

Pro použití šablony je nutné z hlavní nabídky zvolit možnost „*Plot by Template*". Zobrazí se stránka s formulářem, ve kterém je nutné zvolit šablonu a časový interval, na kterém chceme šablonu použít. Stiskem tlačítka „*Generate*" vygenerujeme grafy uložené v šabloně. Z nadpisu lze u každého grafu poznat, jaké informace jsou v něm zobrazeny včetně vyjmenování možností filtru, který je v něm aplikovaný. Při prohlížení vygenerovaných grafů přímo na webové stránce lze využít jejich interaktivnosti, kliknutím se různé části se graf dynamicky překresluje. Ukázky grafů z přednastavených šablon budou představeny v dalších kapitolách spolu s vysvětlení jejich významu.

Tato funkcionalita je určena pro lidi, kteří mají v plánu aplikaci používat pravidelně pro získávání dat do různých reportů, které se víceméně příliš nemění v průběhu používání. V rámci metodiky SCRIL to jsou všechny ty procesy, které svou definicí spadají spíše do části z knihovny ITIL, která pro každý proces přímo určuje jednotlivé klíčové ukazatele výkonnosti, definuje jejich důležitost pro správné fungování celého procesu a vybízí Vlastníky procesů k jejich stanovení pro každý proces. U metodiky

SCRIL jsou to jmenovitě tyto procesy: Správa incidentů, Plnění požadavků, Správa změn a Správa releasů.
Přestože se větší míra dokumentace a papírování neslučuje s popisem a fungováním agilního Scrum, a v praxi se tak pravděpodobně ani neděje, v rámci metodiky SCRIL je doporučeno, aby se procesy testovaly a sbíraly se údaje o jejich fungování i této části vývoje, což je nutné ze dvou důvodů. Zaprvé je důležité otestovat a zhodnotit každý proces, který plyne skrze metodiku SCRIL, aby bylo možné co nejlépe zajistit jejich funkčnost a optimálnost. A protože výstupy ze Scrum slouží jako vstupy do dalších částí z ITIL, je vhodné mít přehled o tom, v jakém stavu jsou požadavky v tomto přechodu předávány dále – tedy zjistit v případě neúspěchu slabé místo celého procesního toku. Aby tato činnosti nebyla příliš časově náročná, je použití implementované aplikace více než vhodné.

9.5 Přenositelnost

V tomto případě je aplikace navržena na spolupráci s konkrétním systémem stopujícím tok požadavků, které zpracovává metodika SCRIL – se systémem JIRA. Aplikace je ovšem přenastavitelná i na spolupráci s jinými systémy (v případě použití jinou společností s jinými preferencemi). Data jsou získávány klientem (který se připojí přes REST aplikaci do systému JIRA) a poté uloženy v databázi, ze které probíhá jejich další zpracování. Převod dat z databáze do lépe uchopitelného formátu provádí Model `IssueMapper.php`. Aplikace je přizpůsobena konkrétnímu projektu s konkrétními parametry a proměnnými, které by bylo nutné v případě potřeby přepsat, funkční logika od zpracování dat až po jejich vykreslení zůstává stejná.

Přenositelnost do jiného systému, než ve kterém byla aplikace implementována, bude vyžadovat dodržet strukturu architektury Model-View-Controler, kterému je aplikace ve spolupráci s frameworkem Zend přizpůsobena. Pro korektní zobrazení grafů bude nutné importovat knihovnu `Highcharts.js` a zdrojové kódy šablonovacího systému Smarty, stejně jako knihovnu jquery.

9.6 Příklady použití

Pro lepší demonstraci použití aplikace v praxi a tím i podpoření navržené metodiky SCRIL bych ráda uvedla příklady použití aplikace v konkrétních případech procesů z metodiky, u kterých si myslím, že je použití podstatné a usnadňuje práci těm, kteří ji používají. (pozn. logika příkladů, které budou dále zmíněny je napasovaná na konkrétní příklad implementace aplikace v prostředí společnosti AVG Technologies, v případě implementaci v jiném prostředí je nutné příklady uzpůsobit, popř. přidat další vhodné příklady).

Proces Správa incidentů, Plnění požadavků a Správa změn (původně ITIL)

Příklad 1.: Roční (měsíční) rozložení úspěšně implementovaných a vydaných incidentů/požadavků/RfC.

Příklad 2.: Kolik ze všech implementovaných požadavků spadá do procesu Správy incidentů/Plnění požadavků/Správy změn.

Příklad 3.: Kolik ze všech zadaných incidentů/požadavků/RfC vyžadující implementaci byly bez problému (na první pokus) vydané do produkce.

Příklad 4.: Kolik ze všech zadaných incidentů/požadavků/RfC vyžadujících implementaci bylo nejvyšší priority.

Proces implementace požadavků (Scrum)

Příklad 1.:	Rozložení vydaných implementovaných požadavků v rámci produktů/programů společnosti.
Příklad 2.:	Kolik požadavků v rámci sprintů byl tým schopen implementovat a vydat.
Příklad 3.:	Rozložení zpracovaných požadavků v rámci sprintu mezi jednotlivé členy týmu.
Příklad 4:	Rozložení všem implementovaných požadavků podle jejich typu.

Proces Správa releasů (původně ITIL)

Příklad 1.:	Kolik releasů bylo vydáno ve vyhrazeném čase.
Příklad 2.:	Kolik releasů se nepodařilo vydat.
Příklad 3.:	Srovnání vydání jednotlivých typů releasů.
Příklad 4:	Kolik bylo vydáno požadavků ve spojitosti s typy releasů.

Vybrané příklady s vygenerovanými grafy a komentáři k získaným výsledkům bohužel z důvodu výskytu interních dat ve veřejné verzi práce nejsou viditelné.

9.7 Další vývoj aplikace

Po dokončení byla aplikace představena ve společnosti AVG Technologies lidem, kteří projevili zájem o její používání, což byl především Release manager a Scrum masteři z jednotlivých týmů. Ukázalo se, že z původního záměru, který byl jednoduché vykreslení grafů pro vložení do pravidelných výsledných zpráv, by se mohla časem stát aplikace, která umožní generovat kompletní reporty s rozsáhlejšími a složitějšími možnostmi úprav grafů, vkládáním mezi ně úseků textů a exportem do pdf formátu. Součástí by mohl být i editor vygenerovaných grafů s jejich dynamickým překreslováním a přidáním dalších projektů do možnosti vykreslování. Dodatečné požadavky byly vzneseny až nějakou dobu po dokončení aplikace a bohužel nestihly být zapracovány do doby před odevzdáním diplomové práce tak, aby mohly být její součástí.

10 Využití metodiky SCRIL v praxi

V Kapitole 8 jsme si popsali navrženou metodiku SCRIL, která vychází ze spojení agilní metodiky Scrum a procesní knihovny ITIL. Metodika byla navržena inspirací situace ve společnosti AVG Technologies, kde se již přes půl rok pro implementaci požadavků na služby používá právě metodika Scrum a kde jsou více než dva roky používány procesy z procesní knihovny ITIL. Popisy jednotlivých procesů v AVG Technologies jsou popsány v Kapitole 7, která byla zpracovávána v lednu 2014. V této kapitole si popíšeme srovnání navržené metodiky SCRIL se situací v AVG Technologies (která se během poslední 5 měsíců zase o kus vyvinula) a zhodnotíme možnosti použití SCRIL v praxi spolu s navržením možných vylepšení.

10.1 Průběh integrace v AVG

Ve veřejné verzi práce tato kapitola z důvodu výskytu interních dat bohužel není dostupná.

10.2 SCRIL v praxi

V této kapitole se budeme věnovat zhodnocení návrhu metodiky SCRIL a především možnosti jejího zavedení do praxe. Návrh byl inspirován konkrétním případem společnosti AVG Technologies, přesto byl navržen natolik obecně, aby se dal aplikovat vhodnou implementací na kteroukoliv jinou společnost, která se zabývá poskytování služeb zákazníkům na denní bázi. Předpoklady pro použití metodiky SCRIL nejsou výrazně omezeny, hlavním a nejdůležitějším předpokladem je mít dostatek financí a opravdu chtít a nechat umožnit vykonat všechny změny, které jsou pro její zavedení potřebné. Před zavedením se ani nepředpokládá přítomnost jedné z obou metodik, nicméně pokud ano, je to výhoda.

Metodika SCRIL popisuje hlavně ty procesy z obou metodik, které jsou podstatné pro jejich integraci. Mimo ně ovšem existuje celá řada dalších procesů, které je třeba vhodně napojit na popsané procesy z metodiky SCRIL. Není nutné, aby byl u všech procesů přesně dodržen jejich popis, nicméně je doporučeno nevynechat žádný ze zmíněných v metodice SCRIL, aby bylo dosaženo nejlepšího výsledku. Stejně jako u jiných metodik, pro konkrétní společnost s konkrétními požadavky a možnostmi je nutné metodiku vhodně implementovat a přizpůsobit.

10.2.1 Srovnání SCRIL s AVG

Ve veřejné verzi práce tato kapitola z důvodu výskytu interních dat bohužel není dostupná.

10.3 Přínosy metodiky SCRIL do praxe

Po navržení byla metodika prezentována vedení CSE ve společnosti AVG Technologies. Protože se jedná o školní dílo a ještě nevyzkoušenou metodiku s návrhem založeným na pozorování a prostudování literatury, nebylo její nasazení povoleno především z důvody možné značné finanční ztráty, nemluvě o nutných organizačních a dalších změnách. Přesto popsané procesy nebyly opomenuty a jednotlivé grafy budou po lehkých úpravách a přizpůsobení potřebám společnosti použity jako interní popisy procesů −

jako lehké úpravy je v tomto smyslu myšleno především zkonkretizování částí procesu a přizpůsobením dalším podrobnostem ve společnosti. Největší využití našel (*Obrázek 25*), který byl umístěn na interní portál oddělení CSE pro názorné vysvětlení, jak funguje Scrum a kde se v procesním toku požadavků nachází.

Navržená aplikace pro tvorbu grafů byla představena Team leaderům a byla přizpůsobena jejich požadavkům tak, aby ji mohli jednoduše používat a maximálně jim usnadnila práci.

Přestože metodika nebyla nasazena na vyzkoušení do praxe, její návrh byl zkonzultován s odborníky z praxe, se kterými byly probrány i potenciální problémy, které návrh přináší. Obecně však byla metodika zhodnocena jako použitelná v případě jejího přizpůsobení konkrétní společností s konkrétními požadavky. Její použití je bráno jako úvodní návrh rozložení procesů pro společnosti, které se chystají používat Scrum spolu s ITIL.

11 Závěr

Cílem této práce bylo zjistit, zda je možné propojit agilní metodiku Scrum a procesní knihovnu ITIL ve společnosti dodávající IT služby zákazníkům a poté ověřit tuto možnost v konkrétní vybrané společnosti.

Nejprve jsem v této práci popsala obě metodiky vývoje software, agilní metodiky zaměřené především na Scrum a poté i procesní knihovnu ITIL a její důležité procesy. Podle nalezených informací z literatury a konzultacemi z praxe jsem zjistila, že spojení obou metodik je možné, protože se jejich procesy vzájemně nevylučují a dají se propojit, přestože se jedná o dva různé přístupy.
Ve druhé fázi jsem zanalyzovala a popsala procesy, které jsou zavedené ve společnosti AVG Technologies, ve spojitosti s integrací metodiky Scrum a ITIL. Nejprve byla popsána jednotlivá oddělení v rámci oddělení CSE, které před rokem zavedlo do vývoje metodiku Scrum s cílem zefektivnit dodávání implementací. Poté byly popsány procesy z obou metodik spolu s důležitými rolemi a artefakty a následně srovnány před a po zavedení Scrum. Zaměřila jsem se především na popis propojení jednotlivých aktivit skrze více oddělení a s dalšími procesy.

Na základě zjištěných informací a problémů, které v AVG během integrace nastaly, jsem v další fázi práce navrhla metodiku, pojmenovanou SCRIL, která se snaží co nejlépe a obecně popsat procesy, které se účastní spojení metodiky Scrum a ITIL. U každého procesu jsem stanovila důležité aktivity a role, které jej ovlivňují, a graficky jsem jej popsala pomocí standardu BPMN. Mimo procesy jsem navrhla a popsala i artefakty (např. Produktový backlog, Produktový release backlog, Operativní backlog, Operativní release backlog apod.), jejichž přítomnost, resp. nepřítomnost může ovlivnit úspěch nasazení metodiky SCRIL do praxe. Po navržení byla metodika SCRIL prezentována a konzultována ve společnosti AVG Technologies. Bohužel použití v AVG nebylo umožněno kvůli možným finančním následkům a z důvodu politiky firmy. Jednotlivé procesy však byly s vedením projednány a bylo poukázáno na možné problémy v mém návrhu. U každého zjištěného problému (např. vhodné složení scrumových týmů, role Vlastníka produktu, vývoj operativních položek) bylo diskutováno jeho nejlepší možné řešení, aby využití celé metodiky bylo co nejefektivnější.

Návrh metodiky jsem podpořila implementací webového nástroje, který byl integrován do interní aplikace společnosti AVG Technologies. Její návrh byl maximálně přizpůsoben reálným potřebám uživatelům v AVG. Aplikace umožňuje kromě vygenerování jednoho grafu podle nastavených parametrů i uložení šablony, ve které budou uloženy parametry pro více grafů, které dotyčný může potřebovat generovat častěji. Aplikace byla navržena tak, aby co nejvíce pomohla podpořit spojení Scrum a ITIL tím, že bude sledovat výsledky, ze kterých lze vyhodnotit, jestli jsou nastavené procesy efektivní, či nikoliv. Je určena především Team leaderům a těm, kteří potřebují získávat jakékoliv výsledky ze svého oddělení a stanovovat na jejich základě další dopady nebo podávat zprávy vedení.

Přestože metodika nemohla být vyzkoušená přímo v praxi jejím zavedením do konkrétní společnosti, byl její návrh diskutován a přizpůsoben připomínkám. Návrhy vizualizací procesů ze SCRIL byly v AVG využity pro tvorbu popisů interních procesů po integraci Scrum. Oproti původním návrhům ze SCRIL byla přidána větší míra detailu u důležitých částí, které byly přizpůsobeny vzhledem ke společnosti. Největší přínos metodiky SCRIL je tak především v popsání v ní obsažených procesů, které prozatím jako spojení ITIL a Scrum veřejně nebyly popsány ani prezentovány a dají se využít jako základní návrh pro popis procesů pro společnosti, které chtějí ve svém vývoji použít metodiku Scrum společně s procesy z ITIL.

Jako hlavním postupem vývoje metodiky SCRIL do budoucna, je její vyzkoušení v praxi u konkrétní firmy. Následným pozorováním a vyhodnocováním efektivity procesů, s využitím aplikace pro tvorbu reportingu, zjišťovat problémy a do metodiky následně zapracovat jejich vyřešení.

Literatura

[1] BUCKSTEEG, M.: *ITIL 2011*. 1. vyd. Brno: Computer Press, 2012, 216 s. ISBN 978-80-251-3732-1.

[2] CARTLIDGE, A. et al (překlad HUDEC J.).: *Úvodní přehled ITIL V3*. Praha: Hewlet-Packard s.r.o., 2007. 56 s. ISBN 0-95551245-8-1.

[3] COCKBURN, A.: *Agile software development*. Vyd. 1. Boston: Addison-Wesley, 2002, 278 s. ISBN 0-201-69969-9.

[4] FARA, M.: *Nástroje pro modelování podnikových procesů*. Bakalářská práce. VŠE, Praha, 2010.

[5] HIGHSOFT AS. *Highcharts* [online]. 2013 [cit. 2014-05-11]. Dostupné z: http://www.highcharts.com/

[6] *ITIL continual service improvement*. 2nd ed. London: TSO, c2011, xi, 246 s. ISBN 978-0-11-331308-2.

[7] *ITIL service design*. 2nd ed. London: TSO, 2011, xi, 442 s. ISBN 978-0-11-331305-1.

[8] *ITIL service operation*. 2nd ed. London: TSO, 2011, xi, 370 s. ISBN 978-0-11-331307-5.

[9] *ITIL service strategy*. 2nd ed. London: TSO, 2011, xii, 483 s. ISBN 978-0-11-331304-4.

[10] *ITIL service transition*. 2nd ed. London: TSO, 2011, xii, 347 s. ISBN 978-0-11-331306-8.

[11] KADLEC, V.: *Agilní programování: metodiky efektivního vývoje softwaru*. 1. vyd. Brno: Computer Press, 2004, 278 s. ISBN 80-251-0342-0.

[12] KOČÍ, R., KŘENA, B.: *Úvod do softwarového inženýrství*. Brno: VUT FIT, studijní opora, 2007, 97 s.

[13] KVĚTOŇOVÁ, Š.: *Strategické řízení informačních systémů*. Brno: VUT FIT, studijní materiál, 2013.

[14] LARMAN, C.: *Agile and iterative development: a manager´s guide*. Boston: Addison-Wesley, 2004, xiv, 342 s. ISBN 0-13-111155-8.

[15] *Manifest Agilního vývoje software*. [online]. 2001 [cit. 2014-01-04]. Dostupné z: http://agilemanifesto.org/iso/cs/

[16] NENADÁL, J.: *Měření v systémech managementu jakosti*. 2. vyd. Praha: Management Press, 2004, 132 s. ISBN 80-7261-110-0.

[17] OMG: *Business Process Model and Notation (BPMN): ver. 1.2*. [online]. 2009 [cit. 2014-01-07]. Dostupné z http://www.omg.org/spec/BPMN/1.2/

[18] RYCHLÝ, M.: *Návrh a implementace IT služeb*. Brno: VUT FIT, studijní materiál, 2013.

[19] Scrum Guide. *Scrum.org* [online]. 2014 [cit. 2014-01-04]. Dostupné z: https://www.scrum.org/Scrum-Guide

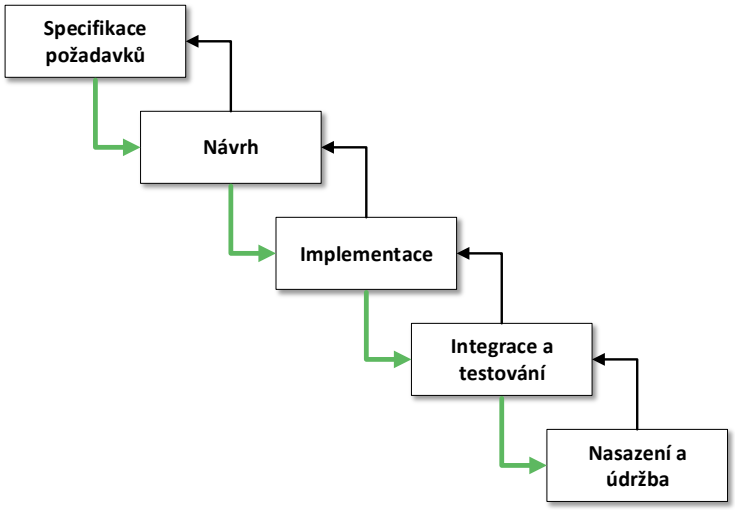
[20] SCHWABER, K., BEEDLE, M.: *Agile software development with Scrum*. Upper Saddle River: Prentice Hall, c2002, xvi, 158 s. ISBN 0-13-067634-9.

[21] *The Free On-line Dictionary of Computing.* [online]. 2013 [cit. 2014-01-04]. Dostupné z: http://dictionary.reference.com/browse/methodology

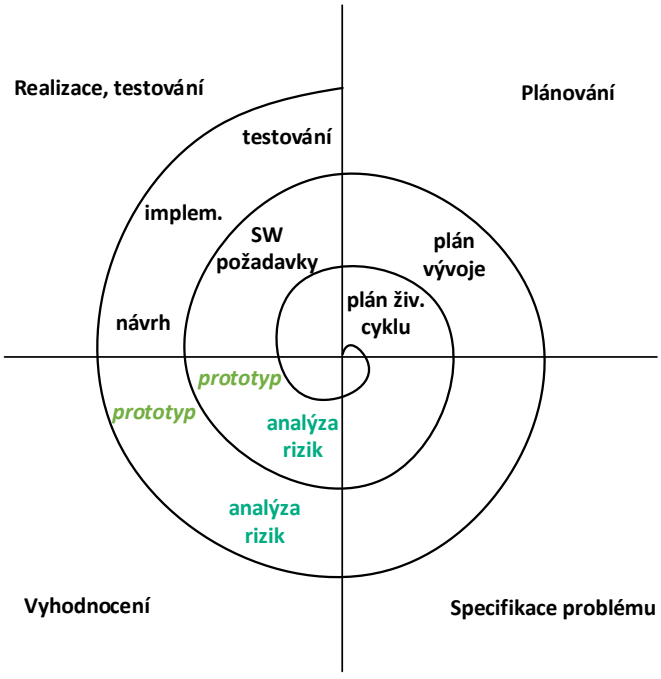
[22] *Všeobecná encyklopedie ve čtyřech svazcích*. Vyd. 1. Praha: Nakladatelský dům OP, 1996-, 4 sv. ISBN 80-85841-17-7.

Seznam příloh

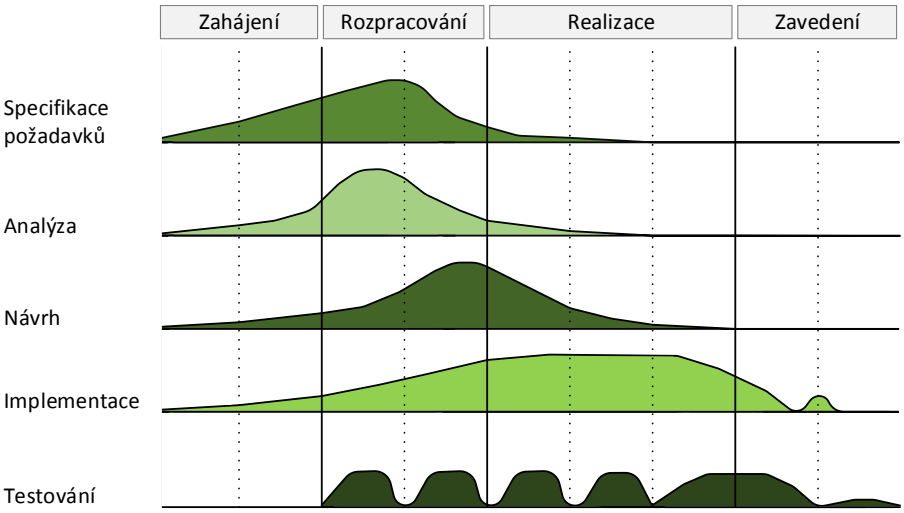
Příloha A. Obrázky



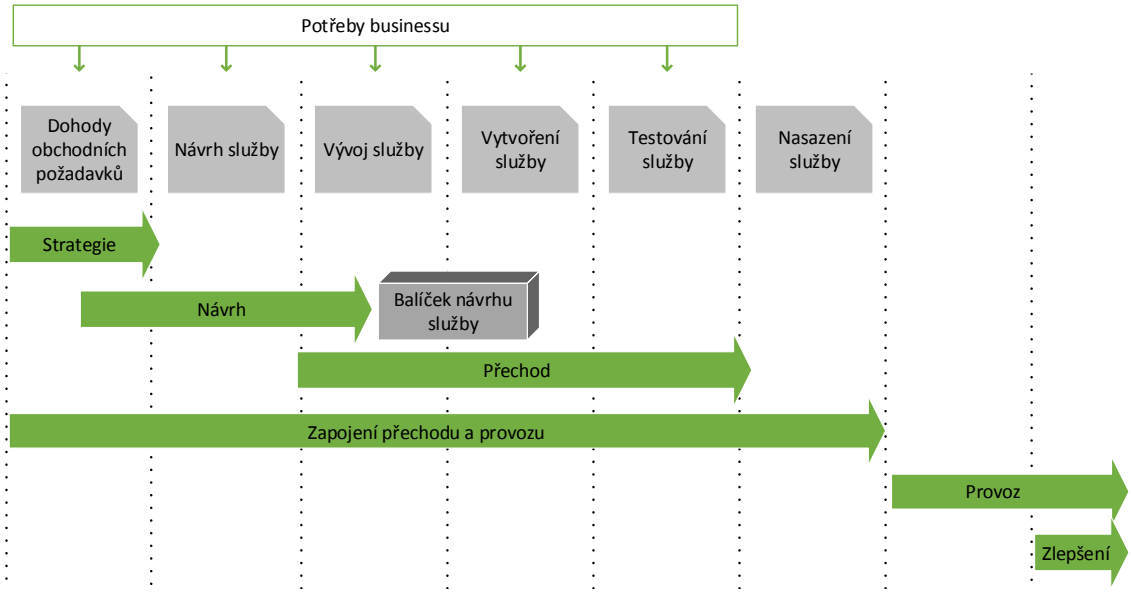
Obrázek 26: Struktura Vodopádového modelu −kromě zelených šipek, znázorňujících hlavní tok přechodů mezi fázemi, zobrazuje i černé šipky, které ukazují možnost návratu o jednu fázi zpět, např. v případě, že při implementaci vývojový tým zjistí, že návrh není správný, lze jej upravit, avšak s tím, že všechny předchozí fáze musí být vykonány znovu.



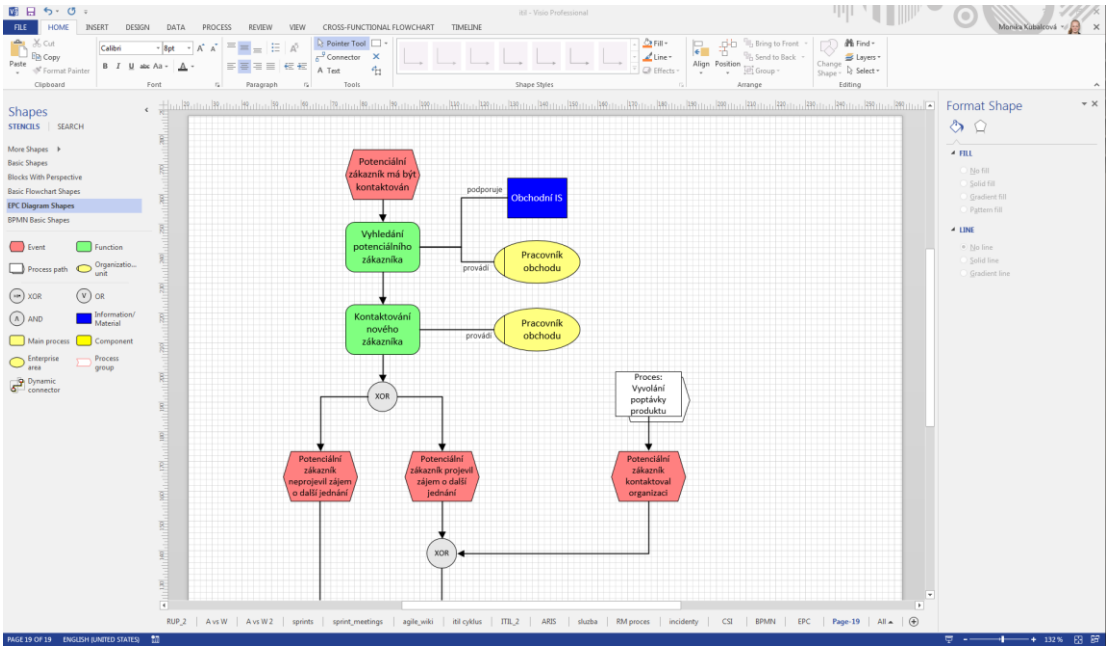
Obrázek 27: Struktura Spirálového modelu



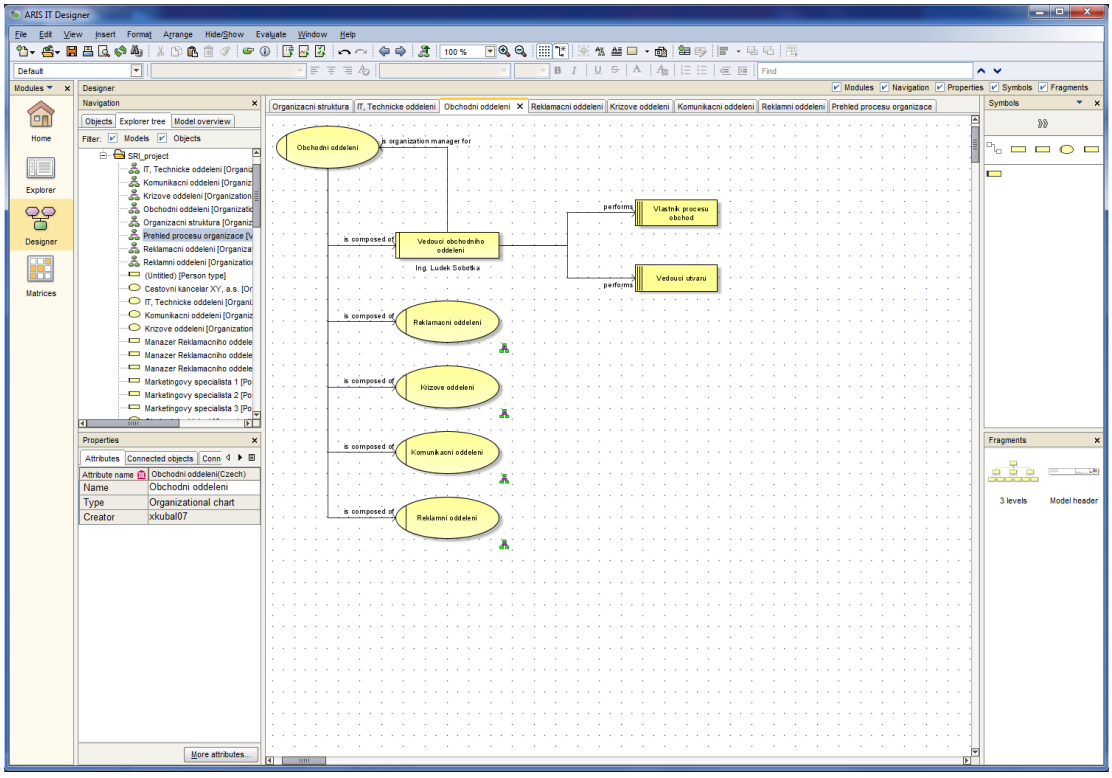
Obrázek 28: Struktura Metodiky RUP



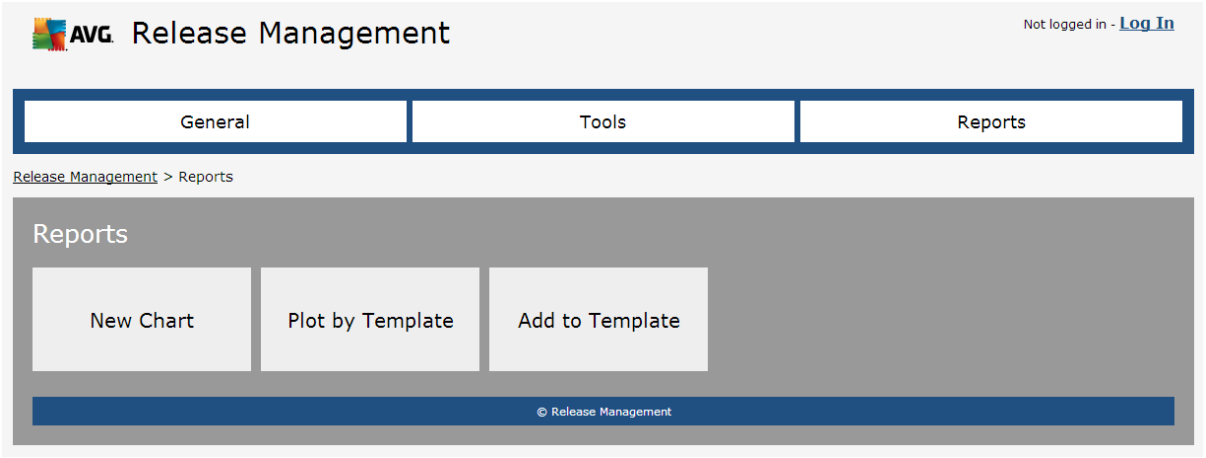
Obrázek 29: Životní cyklus návrhu služby



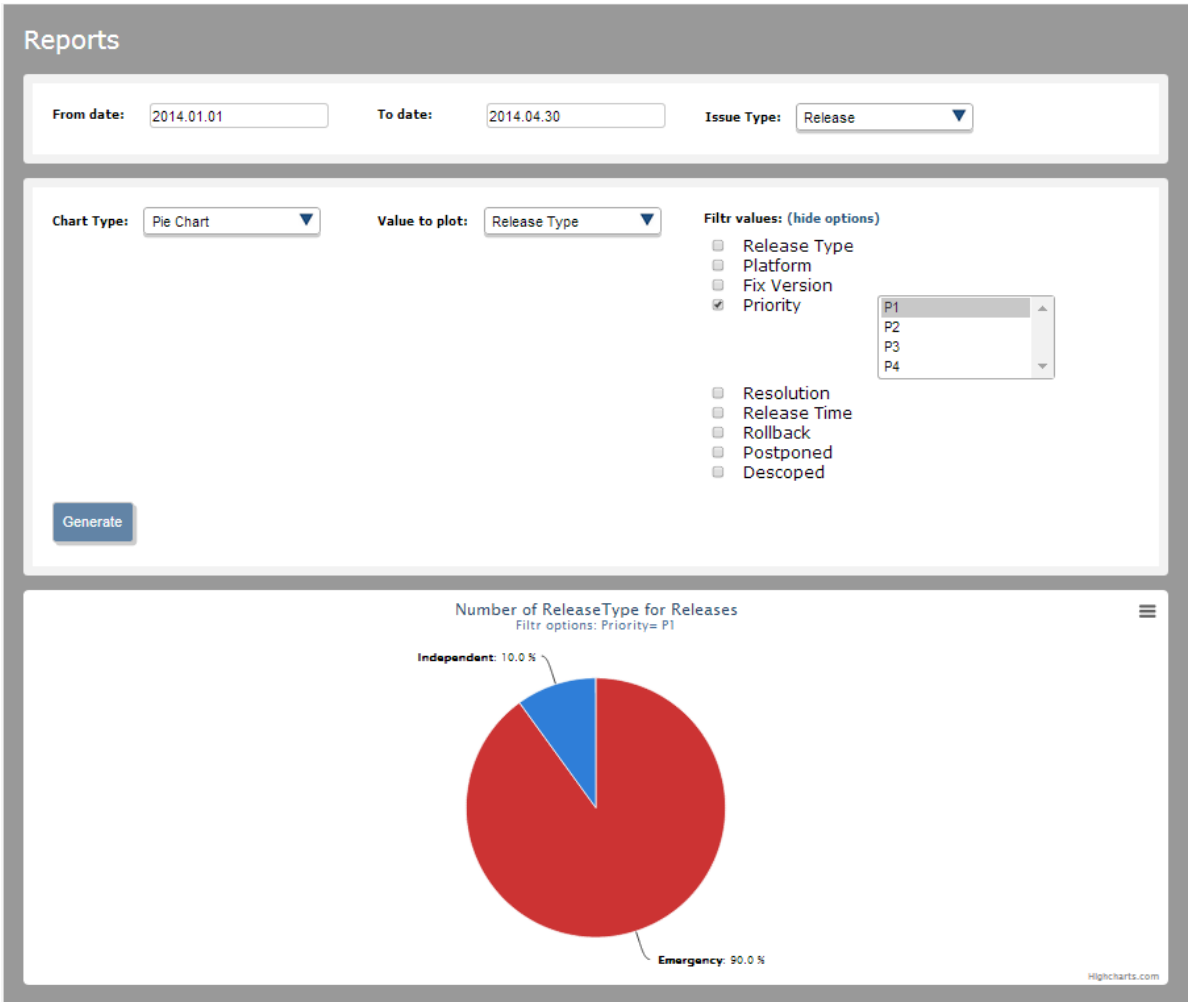
Obrázek 30: Příklad vytvořeného EPC diagramu v nástroji MS Visio



Obrázek 31: Příklad vytvořeného organizačního diagramu v nástroji ARIS



Obrázek 32: Hlavní stránka aplikace pro reporting



Obrázek 33: Formulář pro zvolení parametrů pro filtr spolu s výsledným grafem

Příloha B. Tabulky

EXTRÉMNÍ PROGRAMOVÁNÍ	
Postup	Popis
Plánovací hra	Cílem je naplánovat rozsah pro další vydávání. Účastní se všichni členové týmu a zákazník. Ten vytváří karty zadání, které nesou informace o požadovaných. Datum vydání volí zákazník podle priorit jednotlivých karet.
Malé, časté verze	Verze se vydávají často v menších dávkách, které obsahují funkční část, která je jako celek kompletní a přináší hodnotu zákazníkovi.
Metafora	Představuje v XP jakýsi abstraktní návrh celého systému, který říká jak má systém fungovat. Je sdílená se všemi, kteří se účastní vývoje nebo s ním mají co dočinění (vývojový tým, management, zákazník atd.)
Jednoduchý návrh	Podporuje možnost snazší úpravy návrhu při změně požadavků, nejlépe by měl být tak minimální, aby byl funkční v dané konfiguraci. Další části jsou do návrhu přidány ve chvíli, kdy jsou skutečně potřeba. Může zabránit duplicitám v kódu, obsahuje minimum tříd a metod a je velmi srozumitelný.
Testování	Díky orientaci XP na zdrojový kód je testování velmi důležité a představuje jedinou možnost, jak zjišťovat chyby a zabránit tak špatné produkci. Všechny funkce musí bezchybně projít automatizovanými testy jednotek a funkcionálními testy, které sepisuje zákazník. Programátoři testy jednotek sepisují pro většinu kódu, někdy dokonce před samotným psaním kódu.
Frekventovaný refaktoring	V XP představuje čištění zdrojových kódů a návrhu bez změny funkcionality s cílem zajistit minimální, jednoduchý, srozumitelný kód. Činnost refaktoringu též bývá označovaná jako nepřetržité zlepšování návrhu.
Párové programování	Zdrojový kód je vytvářen vždy párem programátorů, kteří mají jen jeden počítač. Ten z nich, který sedí za klávesnicí a sepisuje kód, přemýšlí nad implementací zvolené metody, druhý z nich přemýšlí nad použitím metody s ohledem na globální strategii, vhodných testech apod. Pozice programátorů se pravidelně střídají.
Společné vlastnictví kódu	Kdokoliv může změnit jakoukoliv část kódu v systému v kteroukoliv dobu. Všichni tedy přebírají zodpovědnost za celý systém a ten, kdo najde místo pro vylepšení je za něj dále zodpovědný.
Nepřetržitá integrace	Všechen kód je nepřetržitě znovu sestavován a testován na odděleném počítači vyhrazeného pouze pro integraci, automatizovanými nástroji.
Udržitelné tempo	XP nepodporuje dlouhodobé přesčasy, protože šťastní, odpočatí programátoři jsou dlouhodobě mnohem efektivnější.
Celý tým spolu	Součástí týmu je po celou dobu vývoje i zákazník, který se účastní vývoje produktu a je nápomocen programátorům při vyjasňování priorit projektu a případným dotazům. Je nutné, aby zákazník byl zúčastněný osobně na místě, popř. skrze vhodné komunikační zařízení.
Standardy pro psaní zdrojových kódů	Protože je zdrojový kód hlavním nositelem informace a mimo něj v XP neexistují žádné další dokumentace a popisy, je nutné, aby programátoři dodržovali určitou štábní kulturu a tvořili pouze dobře čitelný, strukturovaný a srozumitelný kód.

Tabulka 1: Popis postupů pro agilní metodiku Extrémní programování

LEAN DEVELOPMENT	
Pravidlo	Popis
Odstranění zbytečností	Všechny artefakty (dokumenty, specifikace, modely, meziprodukty, atd.), o kterých lze říct, že nic nepřináší výslednému produktu, je vhodné odstranit a při vývoji se s ním dále nezdržovat.
Zminimalizovat zásoby	Aktuálně nepotřebné věci by se neměly vyvíjet a množství dodávaných artefaktů je vhodné omezit, např. dokumentace, o kterých není jisté, že se kdy dostanou do styku se zákazníkem a nepodporují hodnotu výsledného produktu.
Zmaximalizovat tok	Neboli zkrátit čas potřebný pro vývoj produktu, např. rozdělením celého procesu na podprocesy a ty rozdělit mezi různé členy vývojového týmu, aby mohly probíhat paralelně a ušetřilo se tak více času.
Důležitá je poptávka	Odložit rozhodnutí na co nejpozdější chvíli, pokud to situace dovolí, aby v případě změny poptávky trhu nebo zákazníka byl tým schopen reagovat a neplýtval prací, kterou udělal dříve a marně.
Pravomoce pracovníkům	Pro lepší motivaci a docílení radosti z práce řadovým pracovníkům, by jim vedení společnosti mělo dát možnost samostatně rozhodovat o vývojovém procesu.
Vyhovení zákazníkovi	Ideální je zahrnout zákazníka do vývojového procesu a postupně od něj zjišťovat detaily požadavků a diskutovat s ním možné změny s cílem dodat mu produkt, který bude nejlépe korespondovat s jeho představami.
Zpětná vazba	Díky zpětné vazby tým postupně zjišťuje, co je nutné v procesu vývoje změnit, aby vyhovoval požadavkům zákazníka. Před zavedením chyb při změnách v systému je vhodné jej celý otestovat.
Odstranění lokální optimalizace	Pokud optimalizace dílčích meziproduktů nijak nepomůže zlepšit výsledný produkt, je tato činnost považována za zbytečným plýtváním zdrojů.
Partnerství s dodavateli	Je lepší zajistit dodavatele, který kromě dodávaného produktu nabízí i jeho podporu do budoucna, se kterou stoupá kvalita dodávek a tak i kvalita výsledného produktu.
Kultura pro neustále zlepšování	Příjemné pracovní prostředí podporuje pracovníky firmy ke kreativnějšímu myšlení a zlepšení tak kvality výsledného produktu.

Tabulka 2: Popis postupů pro agilní metodiku Lean Development

Příloha C. Obsah CD/DVD

\code	Složka obsahující zdrojový kód aplikace
\application	
\controllers	Složka s kódy pro kontroler pomocí Zend
ReportsController.php	
\models	Složka s kódy pro modely pomocí Zend
Chart.php	Třída pro zpracování informací o grafu
Template.php	Třída pro zpracování informací o šabloně
Reports.php	Hlavní funkce pro dodání dat kontroleru
IssueMapper.php	Kód pro namapování dat z databáze pro další použití
ReportsReleaseItems.php	Uložení dat pro release items z/do databáze
ReportsReleases.php	Uložení dat pro release z/do databáze
ReportsTemplates.php	Ukládání/vkládání/zpracování šablon z/do databáze
ReportsChart.php	Ukládání/vkládání/zpracování grafů z/do databáze
\css	Složka s kódem pro stylování
layout-reports.css	
\img	Složka s obrázky pro aplikaci
arrow.png	
\js	Složka s kódy pro javascript
reports.php.js	Kód pro použití při tvorbě nového grafu a použití šablony
reports-template.js	Javascript pro použití při tvorbě nové šablony
\tpl	Složka s kódy pro vykreslení pomocí Smarty šablon
reports.tpl	Kód pro vykreslení hlavní stránky
new-report.tpl	Kód pro vykreslení stránky s novým grafem
template-new.tpl	Kód pro vykreslení stránky pro tvorbu nové šablony
template-report.tpl	Kód pro vykreslení stránky pro použití šablony
\part\charts	
filter-boxes.tpl	Pomocný vykreslovací kód pro formulář filtrů
chart.tpl	Pomocný vykreslovací kód pro grafy
\scril	Složka obsahující grafy navržené metodiky SCRIL
scril.vsdx	Zdrojový dokument grafů pro metodiku SCRIL
\doc	
xkubal07_dp.docx	Zdrojový dokument textu práce
xkubal07_dp.pdf	Text práce
README	

Příloha D. Návod na instalaci

Jelikož je aplikace implementována jako část interní aplikace společnosti AVG Technologies, jejíž zdrojové kódy nebyly mým autorským dílem a jsou vlastnictvím AVG Technologies, nemohly být tedy přiloženy k výstupům této práce.

Implementovanou aplikaci tak bohužel není možné vyzkoušet bez vložení přiložených zdrojových kódů (s dodržením rozložení adresářů a souborů) do firemní struktury a spustit ji na firemní síti, kde má přístup ke všem potřebným datům.

Funkčnost aplikace tak bude demonstrována autorem práce za použití výpočetní techniky s přístupem do interní sítě AVG Technologies.